1-1-2008

Development Of Face Recognition System Using Verilook Software Development Kit

Paras Pradhan
Eastern Illinois University

DEVELOPMENT OF FACE RECOGNITION SYSTEM USING VERILOOK SOFTWARE DEVELOPMENT KIT

PRADHAN

THESIS MAINTENANCE AND REPRODUCTION CERTIFICATE

TO: Graduate Degree Candidates (who have written formal theses)

SUBJECT: Permission to Reproduce Theses

The University Library is receiving a number of request from other institutions asking permission to reproduce dissertations for inclusion in their library holdings. Although no copyright laws are involved, we feel that professional courtesy demands that permission be obtained from the author before we allow these to be copied.

PLEASE SIGN ONE OF THE FOLLOWING STATEMENTS:

Booth Library of Eastern Illinois University has my permission to lend my thesis to a reputable college or university for the purpose of copying it for inclusion in that institution's library or research holdings.

6/23/08

Author's Signature Date

I respectfully request Booth Library of Eastern Illinois University **NOT** allow my thesis to be reproduced because:

_____

_____

_____

_____ _____

Author's Signature Date

This form must be submitted in duplicate.

DEVELOPMENT OF FACE RECOGNITION SYSTEM USING VERILOOK SOFTWARE DEVELOPMENT KIT

(TITLE)

BY

Paras Pradhan

THESIS

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

Master of Science in Technology

IN THE GRADUATE SCHOOL, EASTERN ILLINOIS UNIVERSITY
CHARLESTON, ILLINOIS

2008

YEAR

I HEREBY RECOMMEND THAT THIS THESIS BE ACCEPTED AS FULFILLING
THIS PART OF THE GRADUATE DEGREE CITED ABOVE

6/20/08
DATE

Peter P. Lin
THESIS DIRECTOR

6-20-08
DATE

DEPARTMENT/SCHOOL HEAD

Thesis Committee

Peter P. Liu

6/20/08

Peter Ping Liu, Ph.D., P.E., OCP, C.Q.E., and CSIT.
Professor
Thesis Director
Graduate Coordinator
School of Technology

Date

6/24/08

Rigoberto Chinchilla, Ph.D.
Assistant Professor
School of Technology

Date

6-20-08

Mahyar Izadi, Ph.D.
Chairman & Professor
School of Technology

Date

ABSTRACT

With built-in libraries and functions, VeriLook Software Development Kit (SDK) provides interfaces for biometric face recognition systems. This research was conducted to gain experience in the development of face recognition system using VeriLook SDK. A prototype named FRECPROJ was developed using the SDK. Major functions were implemented in the prototype, including face enrollment and match using still image files, live streaming with a web camera, or video files.

Through this research, it was realized that a typical face recognition system performs enrollment, verification and identification functions. A single image consisting of a single face is enrolled and the enrolled template is matched with the matching template created from matching image to declare a match or non-match in verification mode. While in identification, images consisting of single or multiple faces per image are used to create a pool of templates and a matching template created from a matching image consisting of a single face is compared with all stored templates to declare a match or non-match. The critical factors for the development of face recognition systems were identified including hardware, operating system, Microsoft Visual C# Express Edition application development environment, VeriLook SDK, referencing libraries and functions of VeriLook from Visual C# environment and developing face recognition software including face detection, template generation, enrolling and matching of faces. Programming techniques were presented in details in terms of implementing the SDK functions in developing the face recognition system. It is noted that the SDK can be utilized to help efficiently develop a face recognition system without being directly involved in the complex algorithms.

ACKNOWLEDGEMENTS

It is a pleasure to thank my thesis supervisor Dr. Peter Ping Liu. With his inspiration and enthusiasm, he helped me to complete this research. Throughout the research, Dr. Liu provided me step-by-step instructions from the project development to thesis structure and grammar correction. I would have been lost without his guidance.

I would like to thank Dr. Rigoberto Chinchilla for his inspiration and support in this research. Dr. Chinchilla is one of my respected faculty members. He has established my background on Biometrics and I have learned a lot from his earnest instructions.

Also I would like to thank Dr. Mahyar Izadi for his valuable tips in this research including aspects of thesis structure and thesis content management.

I wish to thank my wife Srijana for providing a loving environment. I wish to thank my friends Deepak Meriga and Sayed Naveed for the emotional support, entertainment and caring they provided.

Last but not least, I would like to thank all of my professors and friends at EIU for their support from past 2 years.

TABLE OF CONTENTS

Chapter 1

Introduction

Biometric is a branch of computer science and technology, which measures the physical or behavioral characteristics such as fingerprint, face, iris, hand geometry, vein to identify persons (tiresias.org, 2008). "Humans have used body characteristics such as face, voice, gait for thousands of years to recognize each other" (Jain, Ross & Prabhakar, 2004, p.1). Recent advances in computing capability made it possible for automated biometric systems to be effectively used for security purpose. For example, authentication and identification using biometric systems are becoming common in security systems.

Face recognition technology as biometric system is being used for authentication at the present (Ratha, Connell, & Bolle, 2001). Typical face recognition systems use human face to enroll and use the enrolled information for verification and identification. In this technology, facial features are extracted using the information and features of eyes, nose, mouth and jaw edges, which are then stored into the databases for later comparisons in identifying humans (biometricnewsportal.com, n.d). The algorithms extract face data and store only the required minimal information in the form of templates. At the time of verification or identification, the newly acquired face is converted to a template and matched with previously stored template(s).

Various products for biometric face recognition systems are available, ranging from dedicated hardware, software to software development kits (SDK). A SDK is designed for developers and integrators to develop or integrate face recognition system into their own applications utilizing the libraries of SDK (neurotechnologija.com, 2008).

The libraries and functions provided by SDK have capabilities such as camera initialization, extracting facial features from images to create templates and comparing templates.

1.1 Statement of the Problem

One crucial factor in software development and engineering is that it is time consuming. One of the factors that the whole application development cycle depends on is the availability of the resources. Many resources are available for software applications development as database application development and graphics programming. But in the case of biometrics application development, there is a lack of technical resources and support for developers and integrators. The problem that was studied in this research is to solve the complexities of face recognition application development by demonstrating the basic steps on using face recognition SDK libraries and functions for biometric face recognition software application development.

A face recognition application can enroll users, store face image data as templates and use them later for matching purposes. It has capabilities to verify faces from other faces and identify a specific face from multiple faces. Typical face recognition systems initialize camera, extract features from the face, generate a template, store the template in the database or file system with additional information such as name of the person the template belongs to and date of template creation, verify/identify face with the stored templates and also does necessary maintenance such as backing up of templates. To achieve all the functionalities of the face recognition system, advanced functions and face recognition algorithms are required. The face recognition algorithms are based upon complex mathematical representation and computer science topics incorporating those

complex algorithms into a system is very time-consuming. Thus, it is imperative that functions developed extensively can be reused so that the best efficiency can be gained for software development. Therefore, SDK vendors develop libraries and functions that can be reused by the developers and integrators to facilitate development of face recognition systems. A SDK provides developers and integrators with all these functions incorporating the developed algorithms so that developers and integrators themselves do not have to go through the complexities of developing functions and face recognition algorithms.

1.2 Statement of Purpose

A software development kit (SDK) allows programmers to create new face recognition systems or to add face recognition capabilities into their existing applications. However, there are many technical issues to be resolved when using SDK for system development. The main purpose of this research was to study a typical SDK for developing face recognition systems. Issues included:

- Hardware and software system requirements
- Development environment and tools used
- Use and integration of libraries from SDK
- Development of a basic prototype of face recognition application that can enroll users from still images, live streaming video, stored video files and verify or identify persons using facial data
- Identify the basic procedures for the development of face recognition systems

- Analysis of the critical factors associated with the development process and the software used

1.3 Significance of Research

Integrating SDK libraries and functions into the applications allows developers and integrators to add the functionalities of face recognition into the software applications within minimum time and resources. The library and functions in a SDK can be reused in various systems and applications. This research will assist developers and integrators in learning and implementing SDK libraries and functions for application development. Overall, the efficiency of software development and application integration in building face recognition systems was achieved by demonstrating the basic steps and processes in the development aiming to help new developers and integrators in the field of biometric face recognition application development.

1.4 Definition of Terms

Authentication: The technique that confirms the identity of the person who is accessing the system to be known or unknown (bellevuelinux.org).

Authorization: The permitted right or access to use the system (pcmag.com).

Database: Systems that stores the data.

Developers: A compute programmer who develops computer software.

Dynamic link loader (DLL): The library of one or more executable functions that run at the runtime of host application and that cannot be executed directly by the users (webopedia.com).

Integrators: Integrators are the parties who build complete systems using components from different vendors.

Libraries: Routines and programs that can be accessed and reused by the developers in their software development environment.

Open source: Software whose source codes are open to modify.

1.5 Assumptions

- SDK provides developers and integrators with enhanced features for particular application development.
- Developers and integrators are pleased with SDK due to their extending capabilities of functionalities for application development.
- The algorithms built in the SDK are effective for face recognition.
- The library of functions in SDK is reliable and has been tested by the vendor in terms of delivering the functions as designed.

1.6 Limitations

- VeriLook SDK is proprietary and is not open for the modification and customization of face recognition algorithms.
- Currently VeriLook SDK does not conform to some of the face recognition standards such as BioAPI and the template structure provided by the SDK are proprietary.
- VeriLook provides limited number of libraries and functions.
- Functions for face recognition using video files are not implemented.

- Requires web camera capable of streaming video with a minimum resolution of 640x480.
- Requires images having a minimum resolution of 640x480 for enrollment and matching.
- Minimum system requirements include a PC with 1GHZ of processor and 128 MB of RAM.
- Runs only in Microsoft windows, MacOS X and Linux operating systems.

1.7 Delimitations

- Only a demo software prototype was developed.
- The prototype developed was capable of enrolling (including multiple faces from a single image file or live streaming) and matching users based on image files or live streaming video using web camera. In addition, the software is also capable of enrolling and identifying faces from stored video files.
- VeriLook SDK was utilized in the face recognition application development.
- The hardware that were used are Pentium 4 microcomputer with 2.9Ghz Processor speed, 1024 MB of RAM, 160 GB of Hard disk and a Logitech Quickcam web camera with the resolution of 640x480.
- Microsoft windows XP with Service pack 2 was used as an operating system platform.
- C#.NET programming language was used in the development process.
- Microsoft C# Express Edition was used as the development environment.

Chapter 2

Literature Review

Security is a huge concern in this post 9/11 era for every entities ranging from personnel to information systems. "Security is the ability of a system to protect information and system resources with respect to confidentiality and integrity" (Ross, 1999, Computer Security: A Practical Definition, ¶ 2). Weaknesses in security result in the occurrences of unwanted and unexpected situations that can be a tedious task to be resolved needing lots of expensive resources. Hence, the society must take the necessary steps to reduce threats. Various technologies related to security can be implemented to tackle the situations including biometrics systems for facility access controls.

Most of the authentication systems use passwords and/or pins. Biometric systems provide alternatives to them by providing enhanced security. Hence, biometric systems can be used to tighten the security. It has found that systems that are used for immigration purposes, accessing the facilities, logging in to the computer systems use biometrics based authentication and authorization techniques.

2.1 Biometrics

Biometrics can be referred to as authentication and identification techniques that rely on measurable human physical characteristics, which can be automatically checked (webopedia.com). Some biometric identification schemes include fingerprint, face, hand geometry, iris, retina, signature, veins and voice. Biometrics based on gait is still under research (Ronkkonen, n.d) whereas a wide variety of hardware and software systems can be found based on face, finger and iris biometrics.

Biometric systems store information based on physical and behavioral characteristics in form of templates, which are utilized in the matching processes for verification and identification. Benefits include increased security, increased convenience, increased accountability, fraud detection and fraud deterrence (Nanavati, Thieme, Nanavati, 2002). They are widely used in immigration offices, airports and in facility access controls. Moreover, multi biometrics provides enhanced security if implemented appropriately in which two, or more than two, types of biometric authentication systems are implemented, for instance, authentication using finger and face.

2.2 Biometrics Key Terms and Concepts

2.2.1 FAR (False Acceptance Rate)

"A biometric solution's false match rate is the probability that a user's template will be incorrectly judged to be a match for a different user's template" (Nanavati, Thieme & Nanavati, 2002, p.24). False acceptance rate (FAR) or False match rate (FMR) is defined in a situation when an unauthorized person may gain access to the facility. In other words, "if we perform a large number of trials in which people attempt to be authenticated as someone else, the FAR may be thought of as the percentage of time they succeed" (Woodward, Orlans & Higgins, 2003, p.13). Hence, FAR is used to measure the effectives of the biometrics systems.

Mathematically, false acceptance rate is defined as:

$$FAR = \frac{\text{Number of successful fraud attempts against a person}}{\text{Number of all fraud attempts against a person}}$$

FAR can be adjusted using similarity thresholds and scores generated by the matchers.

The terms single FMR and system FMR are also used in the biometric industry. Single FMR can be defined as the false match for a single comparison of two biometrics templates, whereas system FMR is the likelihood of an impostor break-in for the given system considering a person trying to break a system by more than one attempt to match (Nanavati, Thieme & Nanavati, 2002).

2.2.2 FRR (False Reject Rate)

False reject rate (FRR) or False non-match rate (FNMR) is related to the situation when an authorized person is not recognized and denied all privileges. In other words, it is the "rate at which the system incorrectly rejects legitimate matches" (Woodward, Orlans & Higgins, 2003, p.15). FRR can be formulated as:

$$FRR = \frac{\text{Number of rejected verification attempts for a qualified person}}{\text{Number of all verification attempts for a qualified person}}$$

In general, FRR and FAR are inversely related. As FAR is decreased, FRR will be increased. Therefore, they must be calibrated carefully for practical applications.

Single FNMR and system FNMR terms are also used in biometric industry. Single FNMR represents probability of a single user attempt resulting in a false match. Single FNMR doesn't reflect real-world usage. Hence, system FNMR, where a person with more than one attempt for results, denotes system FNMR (Nanavati, Thieme & Nanavati, 2002).

2.2.3 Receiving Operating Characteristics (ROC) Curve

ROC is "a method of showing accuracy of a biometric system" (biometricscatalog.org, Biometrics Glossary, p.24). It illustrates the relation between FMR and FNMR for a system operated using certain threshold values. It is useful to tune the biometric system for FAR or FRR by experimenting the matcher with different threshold values.

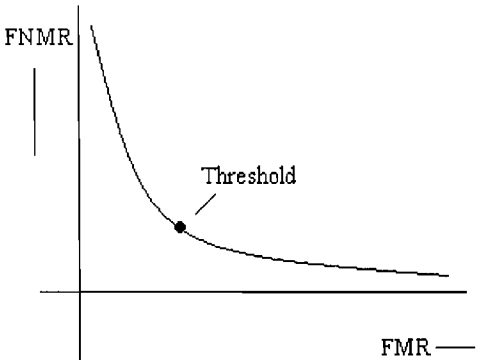

Figure 2.1. ROC Curve

2.2.4 FTE (Failure to Enroll)

When a person is unable to enroll in the biometric system, it is referred to as failure to enroll. FTE occurs when the person using the system has insufficient biometric data. Also, FTE is dependent on the design and policies of the implemented biometric systems. If FTE rate is higher, problematic situation occurs. FTE is measured by Failure enroll rate (FER)(bromba.com, 2008), as follows:

$$FER = \frac{\text{Number of unsuccessful enrollment attempts for a person}}{\text{Number of all enrollment attempts for a person}}$$

2.2.5 ERR (Equal Error Rate)

Equal error rate denotes the overall accuracy of the system. It is an indicator of the system's resistance to break-ins and ability to match templates of authorized users. Known also by crossover rate, it is the rate at which FMR is equal to FNMR (Nanavati, Thieme & Nanavati, 2002).

2.2.6 ATV (Ability to Verify)

Ability to verify is the combination of FTE and FNMR, which denotes overall percentage of users who will be capable of authenticating on a daily basis and is formulated as (Nanavati, Thieme & Nanavati, 2002):

$$ATV = (1\text{-}FTE)*(1\text{-}FNMR)$$

2.2.7 Enrollment (Nanavati, Thieme & Nanavati, 2002)

Enrollment is the procedure to recognize the person's biometric data such as face, iris, retina and fingerprint, and store them in biometric enrollment database. It is basically a learning process to the biometric systems, which is used to collect and store biometric data in the form of small files known as templates.. Typical entities involved in the enrollment process consists of

- Biometric characteristics
- Biometric capture devices
- Biometric feature extractor
- Enrollment database

2.2.8 Templates (Nanavati, Thieme & Nanavati, 2002)

Templates are the stored biometric references for biometric features, which are used for the purpose of comparisons. They are generated at the time of enrollment and biometrics systems utilize the enrolled templates for verification and identification purposes. Templates are small files with sizes varying dependent on vendors. Most templates occupy less than 1 kilobyte of disk space, and most of them are proprietary to each vendor and technology. It should be noted that biometric systems use templates for matching instead of direct images.

2.2.9 Verification (Nanavati, Thieme & Nanavati, 2002)

Verification is the 1:1 (one-to-one) matching process in which the users' data is verified with his/her own enrolled data from the enrolled database and results in a match or no match decision. Verification systems answer the question of "Am I who I claim to be?"

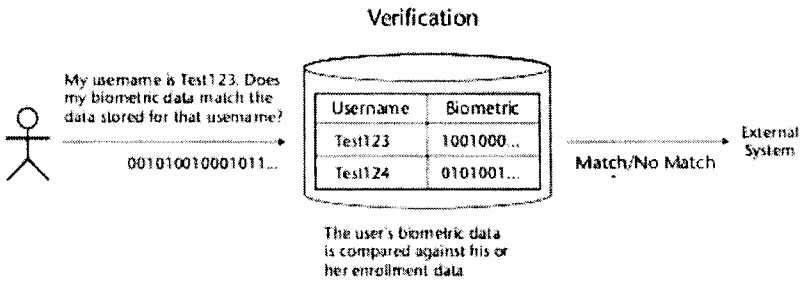

Figure 2.2. Verification System (Nanavati, Thieme & Nanavati, 2002, p.11)

2.2.10 Identification (Nanavati, Thieme & Nanavati, 2002)

Identification is the matching process in which a users' data is matched against a

number of stored enrolled biometric data to find a match. It is also referred to as 1:M matching process, which answers, "Who am I?"

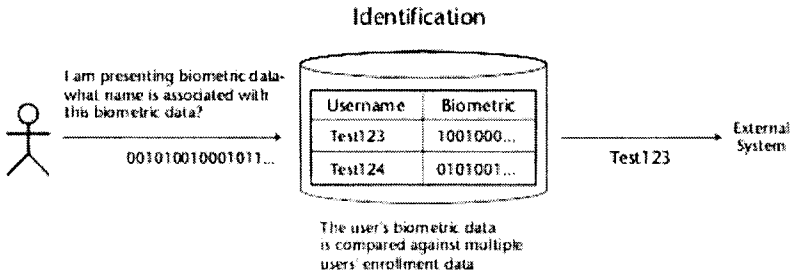

Figure 2.3. Identification System (Nanavati, Thieme & Nanavati, 2002, p.11)

2.2.11 Matching, Score and Threshold (Nanavati, Thieme & Nanavati, 2002)

Matching is a process used to find the degree of similarity between users or templates. The matching process outputs a score, which can be compared with the pre-defined threshold for the declaration of match or no-match. The systems administrator who establishes the degree of correlation necessary for the comparison to be deemed a match generally chooses the threshold. If the matching score exceeds the threshold, it is considered as a match and otherwise a no match.

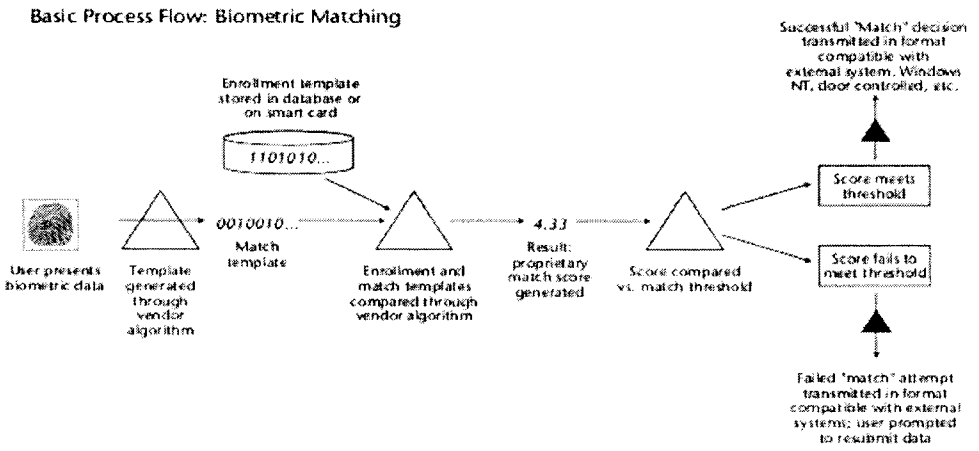

Figure 2.4. Matching process (Nanavati, Thieme & Nanavati, 2002, p.16)

2.3 Biometric Face Recognition

Biometric face recognition is defined as the automated or semi-automated technique for matching facial images (tiresias.org, 2008). Face images have been used for a long time to identify persons. The images consisting of face(s) can be captured from various sources such as image or web camera, analyzed to obtain biometric signatures and are stored as face templates.

Many different proprietary and non-proprietary algorithms exist for features extraction, template generation and matching. Typically most of these algorithms are based upon face scan technologies such as Eigenface, Feature analysis, Neural Network and Automatic face processing (Nanavati, Thieme & Nanavati, 2002).

- Eigenface

  Eigenface is a technology developed at Massachusetts Institute of Technology

(MIT) in which enrollment and verification is achieved using grayscale images. These grayscale images are used to generate templates for enrollment and matching with the information of nose, mouth, eyes and distances between these objects. Multiple eigenfaces are generated from the original image with the means of the mathematical tool called PCA or Principle Component Analysis (Pissarenko, 20021). Eigenfaces are generally the distinctive characteristics features of the face, which represent only certain features of the face in separate ghost like images. The original image can also be reconstructed back using weighted sum of all eigenfaces that contains distinctive features of face. "This weight specifies, to what degree the specific feature (eigenface) is present in the original image" (Pissarenko, 20021, How does it work?, ¶ 2). In the enrollment process, the eigenfaces are mapped into numbers or coefficient to generate templates. During the matching process, a user's template is checked against the enrolled template to determine coefficient variation to declare match or no match. Eigenface technology is basically used for frontal captured images (Nanavati, Thieme & Nanavati, 2002).

- Feature analysis

This technology is based upon eigenface technology but it is more capable of handling changes in appearances such as smiling and frowning. During the enrollment process, feature analysis extracts dozens of features from different locations of the face and also extracts relative location of these features (Nanavati, Thieme & Nanavati, 2002). Feature analysis anticipates that if the features located near the mouth is shifted slightly to another location then it has the smart capability of shifting the location of other adjacent features for proper integrity.

- Neural network

Algorithms based on neural network determine which face features are most effective in face recognition. In this case the features from enrollment image and reference image are processed for a match (Nanavati, Thieme & Nanavati, 2002). If there is a non-match or false match occurring between the faces of the same user, then the system is trained automatically to improve for the matching functionality taking these features into consideration. Hence, the system adjusts them and the matching process is made effective to recognize faces in difficult conditions.

- Automatic face processing

Automatic face processing or AFP is a technology that uses distances and distance ratios between features of the objects such as eyes, nose and corners of mouth, which are easy to acquire (Nanavati, Thieme & Nanavati, 2002). It is not as robust as eigenface, feature analysis and neural network technology, but it is quite useful for the dim lit frontal capture images.

Typical steps in face recognition are (tiresias.org):

- Enrollment
    - Acquiring a sample: In this step face samples are acquired from still image files or a web camera.
    - Features extraction: Facial features are extracted and template(s) is/are generated which is/are then stored in the enrollment database
- Matching
    - Acquiring a sample: The sample image that needs to be matched is acquired using image, video file or web camera.

- Features extraction: Face features are then extracted to generate the template.
- Comparing templates: The template is matched with the stored enrolled template(s) using biometric matcher.
- Declaring a match: Based on the score resulted from the matcher and the defined threshold, a match or no match is declared.

2.4 Biometric Face Recognition Standards

Biometric face recognition standards define the requirements of the face recognition systems needed for interoperability and interchangeability. The major approved biometric standards for the face recognition systems are:

2.4.1 ISO/IEC 19794-5:2005: Information Technology- Biometric Data Interchange Formats - Part 5: Face Image Data (iso.org, 2007)

This standard specifies scene, photographic, digitization and format requirements of images of faces. It defines how a photograph should appear rather than how to take photographs; the image must meet the specification of visible information as gender, eye color, and pose. Moreover, this standard specifies the best practices to capture photos for travel documents.

2.4.2 BS ISO/IEC 19794-5:2005 (bsi-global.com, 2005)

This is the British standard version for ISO/IEC 19794-5:2005 and usability includes identity management systems for document delivery and access, in prison, in

suspecting crime, in citizen rights for voting, unemployment benefits, driving licenses, controlling access to the secure areas.

2.4.3 ANSI INCITS 385-2004: Information Technology-Face Recognition Format for the Data Interchange (nist.gov, 2007)

This standard specifies photographic properties including environment, pose, focus, digital image attributes and face interchange format for relevant applications.

2.4.4 ANSI/INCITS-ITL 1-2000: Information Systems-Data Format for the Interchange of Fingerprint, Facial, Scar Mark & Tattoo (SMT) Information (tilton, 2006)

This standard provides an XML representation for fingerprinting, facial, scar mark and tattoo image data.

2.5 Biometric Face Recognition Technology Vendors

Biometric vendors develop hardware systems, computer software and SDKs for face recognition solutions. Hardware systems are the devices, which can be installed directly into the facility access controls. Face recognition software need to be installed into the computer that is connected with surveillance cameras. The installation of software and accessories must be done by the computer software expertise. Moreover, currently many face recognition SDKs are available that developers may use to create new face recognition software or to integrate face based authentication systems into their existing applications.

Some vendors of biometric face recognition SDKs are (biometricwatch.com, n.d):

Table 2.1. Biometric face recognition vendors

Vendor	Description
XID Technologies	Provides SDKs and solutions for face detection, face recognition, face synthesis, 3D face animation
AcSys Biometrics	AcSys FRS SDK provides tracking, enrollment, verification, classification, database, communication and multimedia controls
Animetrics	Animetrics90 SDK provides accurate face recognition, analysis and visualization from image and video source files
BioID	BioID SDK provides multimodal biometric solutions by face recognition, voice recognition and lip movement recognition
Cognitec Systems GmbH	FaceVACS SDK by Cognitec has basic functions of enrollment, verification, identification and defines abstractions to support user defined applications
Dream Mirh	Mirh Eye SDK offers toolkit for face detection and recognition software systems. It can detect face data by 24Bit RGB from a picture and

	MPEG or a Black and white picture
Identix/L-1 Identity Solutions	FaceIt SDK by Identix provides face finding, quality assessment and 1:1 matching capabilities
Neurotechnologija	VeriLook SDK by Neurotechnologija has capabilities for 1:1 and 1:M matching, simultaneous multiple face detection, processing and identification with a comparison speed of 100,000 faces per second

Besides the proprietary vendors, there are organizations that develop SDK and make their source code available for other users. These are named open source SDK. A few of the open source SDK examples are listed in Table 2.2:

Table 2.2. Open source biometric face recognition SDKs

Product	**Description**
Multimodal Biometric Application Resource Kit (MBARK)	Developed by National Institute of Standards and Technology (NIST), Used to develop multimodal biometric software applications.
OpenCV	Collection of algorithms and sample code for various computer vision problems originally developed by Intel.

2.6 Software Development

Software development utilizes built-in and/or third party functions and libraries according to ones' need. Software Development Kit provides feature rich functions and libraries in software development. Hence by integrating the development tools and software SDK together, the development process will be much simpler rather than going in the direction of complex functions and algorithms development.

The typical software development process involves:

- Requirement analysis
- Design
- Implementation
- Testing and Debugging
- Put in to the operation or deployment
- Maintenance or refinement

API (Application program interface) can be defined as set of routines, protocols and tools for building software applications (querycat.com, n.d). They are also the building blocks that make programming easier. SDK is a programming package consisting of APIs, programming tools and documentations (webopedia.com, n.d). Companies develop SDKs consisting of APIs and distribute/sell to the community for rapid application developments.

2.7 Microsoft .NET Framework, Visual Studio.NET

.NET framework was developed by Microsoft in order to facilitate application development for Windows environment. "Programmers do not have to reinvent the

wheel as the framework provides a rich library of APIs that applications can use" (Bolton, n.d, Definition, ¶ 1). Many built-in libraries preexist in the .NET framework such as libraries for graphical user interface, accessing databases, and networking. The framework can be used to develop different types of windows as well as web-based applications. It supports languages such as C#, C++, ASP, VB and J++.

Visual Studio.NET is a suite of programming languages and related development utilities that runs on the top of .NET framework (pcmag.com, n.d). It is also a product by Microsoft and includes compilers and interpreters for C, C++, C#, BASIC, and J++. Visual Studio needs to be purchased. But, Microsoft has also provided visual studio in a light-weighed version aiming for students and hobbyists in a form of Visual Studio Express Editions, which can be downloaded without cost (microsoft.com, 2007).

2.8 Face Recognition Software Development Kit (SDK)

Face recognition SDK is the collection of APIs, tools and documentation that can be used for the development of biometrics face recognition software. It can also be used to integrate face recognition technology in the user's own applications by themselves without relying on the third party for recognition feature integration. Most of the face recognition SDKs run on various platforms including Windows, Linux, MacOS X and provides application program interface for:

- Camera initiation and management
- Enrollment (using images, live video stream)
- One to One Verification

- One to many Identification
- Other software functions

2.9 VeriLook SDK

VeriLook is the PC-based SDK designed for software developers and integrators developed by Neurotechnologija. It allows rapid development of biometric applications with the help of libraries and functions (neurotechnologija.com, 2008). Information of VeriLook SDK can be found at http://www.neurotechnologija.com/. Major computer industries like Lenovo have adopted SDKs from Neurotechnologija (neurotechnologija.com, 2008).

Some of the features of VeriLook SDK include:

- Face signatures detection from live video and still images
- Designed for 1:1 and 1:M matching modes
- Simultaneous multiple face detection
- Processing and identification of faces with a comparison speed of 100,000 faces per second
- Applications can be developed using programming tools such as C#.NET, VB.NET, Delphi and GNU C/C++ supporting cross platform development and implementation
- Does not require special hardware. Requirements include a simple web camera supporting a minimum of 640x480 image or video resolution, PC with 1 GHZ or better processor, and 128MB of RAM. Supported application development tools include Microsoft Visual Studio, Borland Delphi or Microsoft Visual Studio

- Templates are approximately 2.3 Kbytes

The libraries that VeriLook SDK provides include:

- NCore Library: This library provides infrastructure for Neurotechnologija components.
- NImages Library: This library provides functionality for loading, saving, converting images into various formats.
- CameraMan Library: It provides functions for working with cameras such as camera initialization and capturing current frame.
- VLExtractor library: This library is used to find faces in images and extract features to create templates.
- VLMatcher library: This library is used to match templates with the acquired image.

2.10 Summary

Face recognition systems are gaining more and more applications day by day. Many access controls today are equipped with face authentication systems. Hence, the importance of development of face recognition systems is invaluable. Face recognition SDK eases the complexities of face recognition software systems development by providing ready to use libraries and functions. Hence, this literature provides brief knowledge to biometrics face recognition systems, problems with system development and SDK as a solution to develop face recognition software systems.

Software development is a complex process where several advanced tools and techniques are essential to be implemented in order to generate a workable high quality

product. SDK provides easy interfacing of functions and algorithms for rapid application development with potential savings and improvement of efficiency by reusing of the algorithms. Technical resources such as tutorials, documents and techniques are necessary for a rapid and efficient development. However, there is a lack of resources on using SDKs presently in the existing literature. The research demonstrated the uses of functions, libraries and techniques with VeriLook SDK and it demonstrated ways to integrate the VeriLook SDK functions and library into software applications. It is hoped that the exemplary use of face recognition SDK will help ease the difficulties in accessing and understanding the face recognition SDK for future development.

Chapter 3

Research Methods

Software SDK adds rich functionalities to the development environment. VeriLook SDK offers a library of functions for face recognition system development, and provides interfaces to access the face recognition algorithms (neurotechnologija.com, 2008). It is important to understand how to utilize the libraries and functions for development. The purpose of this research was to determine the procedures and implementation details for creating face recognition systems by integrating VeriLook SDK. A demo prototype of face recognition software was developed to gain realistic experience and verify the understanding of VeriLook SDK.

3.1 System Setup

3.1.1 Hardware

The following computer hardware was used in this research project:

- A Pentium 4 microcomputer with 1GB of RAM, 2.9 GHZ processor and 160 GB of hard disk space
- A web camera capable of streaming video and taking images in the resolution of 640x480 pixels

3.1.2 Operating System Environment

The operating system that was used in this research project is Microsoft Windows XP operating system with service pack 2 and regular updates

3.1.3 Development Environment

- Microsoft .NET 2.0 framework
- Microsoft Visual C# Express Edition
- VeriLook SDK 3.0

3.2 Major Functionalities of the Software

The software prototype that was developed in this research project is able to enroll, verify and identify users based on the input sources including still image files or live streaming from the web camera. Moreover, the program is capable of enrolling and identifying human based on stored video files.

3.2.1 Enroll and Match using Still Images

In order to enroll users with still images, the requirements of the images such as image format and image size for the generation of templates are checked and evaluated. Then the images are converted to gray scale for further processing.

Enrollment of a single face from a still image takes place after a face is detected, its feature extracted, and a template is created. In this case a single image should not contain more than one face.

The matching process of a single face from a single image is based upon the template to be extracted and compared with the enrolled template to declare match or no match.

Multiple face detection from a still image is initiated by detecting number of faces in a single image file. With the help of eyes detection for every face, enrollment process is performed to create templates. One or more faces face(s) are extracted from the image and thus multiple templates may generate in one attempt of enrollment. The matching process compares the newly generated templates with the stored templates to determine match or no match.

3.2.2 Enroll and Match using Live Streaming Web Camera

The requirements of the web camera that is used to capture images from the live video stream is evaluated and checked before enrolling and matching of faces using live video streaming. In this process, image frames are captured from the live streaming and templates were created and stored.

An image consisting of a single face is captured from the live video streaming and stored temporarily. Then the face feature is extracted from the stored image frame. A template is created, stored and the image frame is deleted.

The matching process captures an image from the live video streaming from which the template is generated and compared with the enrolled template to decide a match or non-match.

In the enrollment process, which can generate and store multiple templates in one attempt, the image containing multiple faces is captured and stored from the live video stream from which the biometric extractor generates template(s). In the matching

process the image is captured and the template(s) is/are created to check against the stored templates for the declaration of a match or no match.

3.2.3 Enroll and Match using Stored Video Files

Requirements of video files from which the images are to be extracted are validated in order to enroll and identify faces. In this scenario, the image frames are extracted from a video file using specified media time, which are then used to extract face(s) for comparing with the stored templates to declare match or no match decision.

3.3 Development Procedures

A typical procedure for the development of face recognition software includes:

1. Choosing face recognition SDK (VeriLook was chosen in this research)
2. Choosing appropriate development environment

In this research, since VeriLook SDK can be operated under the lightweight free Microsoft Visual Express Editions, Microsoft Visual C# Expression Edition was chosen. The idea behind choosing C#.NET platform was that code writing complexities are better handled in C# as compared to C++ and other languages. Moreover, complex pointers as in C++ can be avoided in C#, resulting speedy learning curve of the language.

3. Implement basic enroll, verify and identity functionalities for enrollment and matching purposes

A typical face recognition system consists of methods to enroll users, verify/identify users and to control threshold parameters for the match or no match decision. Basic techniques that were implemented for the development of face recognition system in this research project include:

3.3.1 Enroll a Single Face from a Still Image

The Codes that were used to enroll users are as follows: (neurotechnologija.com, 2008)

```
# Loading an image from a file
NImage image = NImage.FromFile(filename);

# Converting image to grayscale as VeriLook requires grayscale image
NGrayscaleImage grayscale = (NGrayscaleImage)NImage.FromImage(NPixelFormat.Grayscale, 0, image)

# Initializing an extractor
VLExtractor extractor = new VLExtractor();

# Declaring variable that carries face detection results such as coordinates of face and eyes
VleDetectionDetails details;

# Template generation
byte[] template = extractor.Extract(grayscale, out details);

# Compressing template file before storing
byte[] compressed = extractor.Compress(template);
```

Where,

filename = Input image file

template = Output template file

extractor = VeriLook extractor which can extract faces from images

details = Information result of face detection such as co-ordinates of face and eyes

grayscale = Image in gray scale format

compressed = Template in compressed format

In order to enroll a single face from a still image file, the image file needs to be open and imported in the process. NImage, a SDK function, accomplishes the above operations that open the image file named "filename." As the next step, the imported image is converted into gray scale since the face extracting algorithms requires that. Then, a face extractor is initiated using "VLExtractor" (SDK function) and a variable for

face extraction details (named "details") is declared. The extractor generates a template from the gray scale image ("grayscale"), which also generates face feature details. Finally, the template is compressed by "extractor.Compress" function. The compressed template is named as "compressed" and stored as a binary type.

3.3.2 Enrolling Multiple Faces from an Image

Codes that were used: (neurotechnologija.com, 2008)

```
# Loading an image
NImage image = NImage.FromFile(filename);

# Converting image to grayscale as VeriLook requires grayscale image
NGrayscaleImage grayscale = (NGrayscaleImage)NImage.FromImage(NPixelFormat.Grayscale, 0, image)

# Initializing extractor
VLExtractor extractor = new VLExtractor();

# Creates array of faces
VleFace[] faces = extractor.DetectFaces(grayscale);

# Detect eyes for every face, create and compress templates with the help of eyes
for (int i = 0; i < faces.Length; ++i)
        {
                VleEyes eyes = extractor.DetectEyes(grayscale, faces[i]);
                byte[] template = extractor.Extract(grayscale, eyes);
                byte[] compressed = extractor.Compress(template);
        }
```

Where,

faces [i] = Array of detected faces

eyes = Structure consisting of information of eyes

DetectEyes = Method to detect eyes in an image

compressed = Template in compressed format

filename = Input image file

template = Output template file

extractor = VeriLook extractor which can extract faces from images

grayscale = Image in gray scale format

To enroll multiple faces from a still image file, the image file needs to be open and imported in the process. NImage, a SDK function, is used to accomplish the above operations that open the image file named "filename." Then, the imported image is converted into gray scale since the face extracting algorithms requires that. As a next step, a face extractor is initiated using "VLExtractor" (SDK function) and an array structure variable "faces" for storing detected faces is declared. The extractor detects the total number of faces and a C# for loop is used to loop through all the detected faces to create templates. To generate face templates, face detection using eyes is used in which the information of eyes are used to differentiate faces. Also, the template is compressed by "extractor.Compress" function. The compressed template is named as "compressed" and is stored as a binary type.

3.3.3 Verification

Codes that were used: (neurotechnologija.com, 2008)

```
# Loading an image
NImage image = NImage.FromFile(filename);

# Converting image to grayscale as VeriLook requires grayscale image
NGrayscaleImage grayscale =
(NGrayscaleImage)NImage.FromImage(NPixelFormat.Grayscale, 0, image)

# Initializing extractor
VLExtractor extractor = new VLExtractor();

# Variable declaration that carries face detection results such as coordinates of face and eyes
VleDetectionDetails details;

# Template generation
byte[] template = extractor.Extract(grayscale, out details);

# Load the enrolled template to variable enrolledtemplate using File Input/Output
enrolledtemplate = /* Procedure to load stored templates */
# Matcher initialization
VLMatcher matcher = new VLMatcher();
# Compare new template with enrolled template and generating output as score
```

```
double score = matcher.Verify(template, enrolledtemplate);

# Declaring match or non-match based on the  threshold of 0.65
if (maxscore > 0.65)
{
        MessageBox.Show("Match");
}
else
{
        MessageBox.Show("Non-Match");
}
```

Where,

template = New template to be matched

enrolledtemplate = Stored template in the database

score = Output match score between 0 to 1. 0 means no similarity, while 1 is 100% similarity or perfect match

matcher = VeriLook matcher than match templates

filename = Input image file

extractor = VeriLook extractor which can extract faces from images

grayscale = Image in gray scale format

In order to verify a face from an image with the stored template, the image was loaded and converted into a gray scale. Then, a face extractor was initiated using "VLExtractor" (SDK function) and a variable for face extraction details (named "details") was declared followed by a template creation. As a next step, matcher was initiated and the enrolled template was also loaded. Finally, a matching score was generated using "matcher.Verify" function that compares the both templates. The output score was then used to declare match or no match with the reference of threshold value.

3.3.4 Identification

Codes that were used: (neurotechnologija.com, 2008)

```
# Loading an image
NImage image = NImage.FromFile(filename);

# Converting image to grayscale as VeriLook requires grayscale image
NGrayscaleImage grayscale =
(NGrayscaleImage)NImage.FromImage(NPixelFormat.Grayscale, 0, image)
# Initializing extractor
VLExtractor extractor = new VLExtractor();

# Variable declaration that carries face detection results such as coordinates of face and eyes
VleDetectionDetails details;

# Template generation
byte[] template = extractor.Extract(grayscale, out details);

# Matcher initialization
VLMatcher matcher = new VLMatcher();

# Initialization of template which is matched against all the stored templates
matcher.IdentifyStart(template);

# Load the stored templates into an array storedtemplates using File I/O
storedtemplates[] = /* Procedure to load stored templates*/

# Comparing template with enrolled templates and declaring match or non-match
double maxscore = 0;
for (int i = 0; i < templateCount; ++i)
{
        double score = matcher.IdentifyNext(storedtemplates[i]);
        if (score > maxscore)
      {
        maxscore = score;
        }
}
if (maxscore > 0.65)
{
MessageBox.Show("Match");
}
else
{
MessageBox.Show("Non-Match");
}

# Stop the identification process
matcher.IdentifyEnd();
```

Where,

template = Template that needs to be matched

IdentifyStart = Initializing the template

templatecount = Number of templates stored in the database

storedtemplates = Stored templates database

IdentifyNext = Method to move the pointer to next templates

score = Output match score between 0 to 1. 0 means low or no similarity, while 1 is 100% similarity or perfect match

filename = Input image file

extractor = VeriLook extractor which can extract faces from images

grayscale = Image in gray scale format

In order to identify a face from the stored templates, the image is loaded and converted into a gray scale. Then, a face extractor is initiated using "VLExtractor" (SDK function) and a variable for face extraction details (named "details") is declared followed by a template creation. As a next step, matcher is initiated, initialized with the created template and also the enrolled templates are also loaded into an array. Then, using the C# for loop and "matcher.IdentifyNext" SDK function, the template is compared against all the stored templates. Matching scores are generated in the process. If any one among the generated scores is greater than the threshold value, then it is declared as a match otherwise a no match.

3.3.5 Working with Web Camera

Code that was used: (neurotechnologija.com, 2008)

```
activeDevice.GetCurrentFrame().Save(@"c:\\frecproj\\frecdata\temp_images\ver_enroll.jpg");
```

An image is captured from a live streaming web camera, which is then used to enroll and match face(s) using images. A SDK function "activeDevice.GetCurrentFrame" is used to capture the current running frame and store as a JPEG image file.

3.3.6 Working with Video Files

In this scenario, images are extracted from video files using ImediaDet interface provided by Microsoft and built-in to the Microsoft based operating systems, which is intended to work with video files such as for extracting image frames and gather video file information. The images are extracted from the video at the specified media time using C# for loop, converted to JPEG format and then are used to enroll and identify users.

Codes that were used: (JockerSoft, 2006 & neurotechnologija.com, 2008)

```
# Initialize IMediaDet
MediaDetClass md = new MediaDetClass();

# Loading video file (Avi or WMV)
md.Filename =  filename;

# Extracting and saving frames in JPEG file format (10 frames or 1 to 10 seconds of a video file in this
example)
for (int i = 0; i < 10; i++)
        {
                string fBitmapName = i + "-sample" + ".bmp";
                md.WriteBitmapBits(i, 640, 480, fBitmapName);
                Image img = Image.FromFile(fBitmapName);
                img.Save(@"c:\\" + i + "-sample" + ".jpg", ImageFormat.Jpeg);
        }
```

Where,

md = MediaDet class

fBitmapName = Bitmap filename to be created from video

i = Current frame of the video

After the images are extracted from video files, templates are created from every image. To remove duplicate templates of same face, the first templates are compared with other templates. If a score was greater than a threshold, it is considered as a repeating template. The repeating templates are deleted.

In identification using video files, frames are extracted from video files and templates are generated from the extracted images. Also in this case, repeating templates of a same face is/are removed by utilizing VeriLook matcher and the generated score. As a next step, the filtered templates are compared against the pool of templates to declare match or non-match.

3.3.7 Controlling Matching Threshold and False Acceptance Rates

The matching threshold values according to the False acceptance rates was studied and implemented into the prototype. The decision was based on the output score generated by the VeriLook matcher.

Codes that were used:

```
if (score > 0.625)
{
        MessageBox.Show("Matched");
}
else
{
        MessageBox.Show("Macth failed");
}
```

Where,

0.625 = Threshold value according to False Acceptance Rate of 1%

(neurotechnologija.com, 2008)

After the completion of prototype development, the development procedure was documented in this thesis, including setting up the SDK and Visual C# Express, technical details on the enrollment and matching of the faces using the image, video stream and video files.

Chapter 4

Results

In this chapter, details are presented on setting up of development environment and basic steps on development of face recognition system that can enroll and match according to the different input sources such as image, live web camera and video files. In addition tips on checking the minimum requirements of input sources are also demonstrated. A prototype named FRECPRO was developed and the steps that are shown in this chapter are based upon it.

4.1 Development Environment Setup

Procedures for installing and configuring the development environment for the development of face recognition system are presented as follows:

4.1.1 Downloading and Installing Microsoft Visual C# Express Edition

VeriLook SDK supports languages such as C#, C++ and Visual Basic. In this research study C# was chosen. Application development in C# can be achieved using Microsoft Visual C#. Microsoft has released C# development environment in different two flavors. They are Microsoft Visual C# as part of Microsoft Visual Studio and a standalone Microsoft Visual C# Express Edition. Microsoft Visual C# has to be purchased while the latter can be downloaded without any charge. I was found that Microsoft Visual C# Express is sufficient to integrate VeriLook libraries and functions in developing face recognition systems.

Basic steps in downloading and installing Microsoft Visual C# Express Edition were identified as:

1. Open the download website at http://www.microsoft.com/express/download/default.aspx .

2. Click the download button in the Microsoft Visual C# 2008 Express Edition section. This will download the installer file vcssetup.exe.

3. Run the downloaded file to install Visual C# Express Edition. Continuous Internet connection is required to install Visual C# Express Edition.

4. Click Next and simply use the default settings on the upcoming windows to install Visual C# Express Edition. The installer will download Visual C# components, .NET framework, some necessary components and install them.

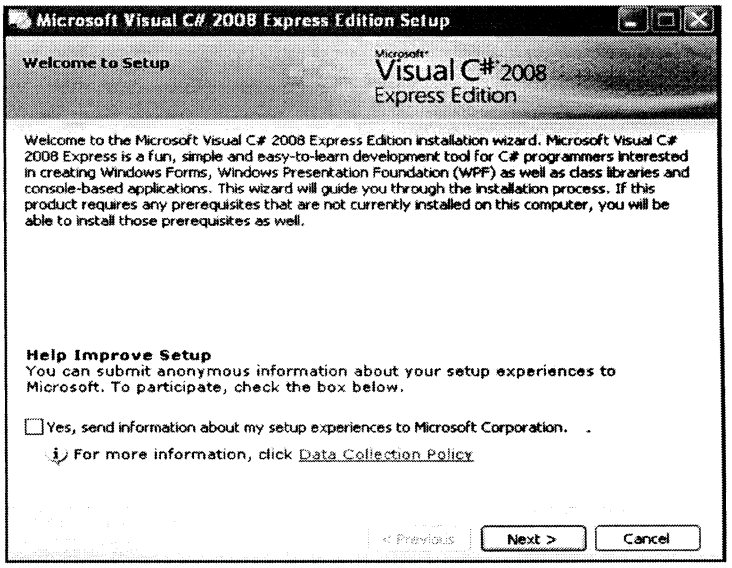

Figure 4.1. Microsoft Visual Studio Express Edition setup screen

4.1.2 Visual C# Application Development: A Simple Example

This section shows a basic application development using Visual C# Express Edition in order to get started with Visual C#. The following steps demonstrate the creation of a "Hello World" application.

1. Start Visual C# Express Edition.

2. Go to File, and click New project.

3. Make sure the "Windows Form Application" is selected. Provide a name to this project (Example: helloproj). A new workspace will be created with a window form.

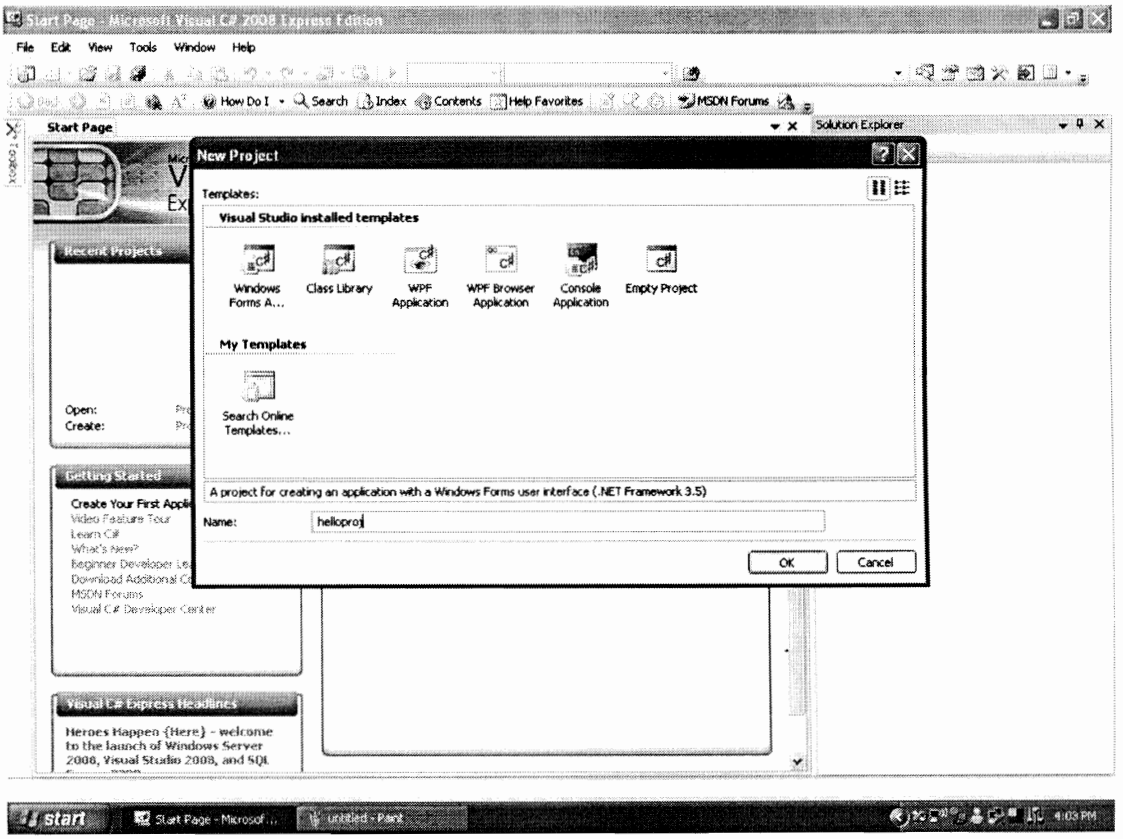

Figure 4.2. Creating a new project in Visual C# Express Edition

4. Click "View → Toolbox" to show the components.

5. Drag a button component and a label component to the window form from Toolbox.

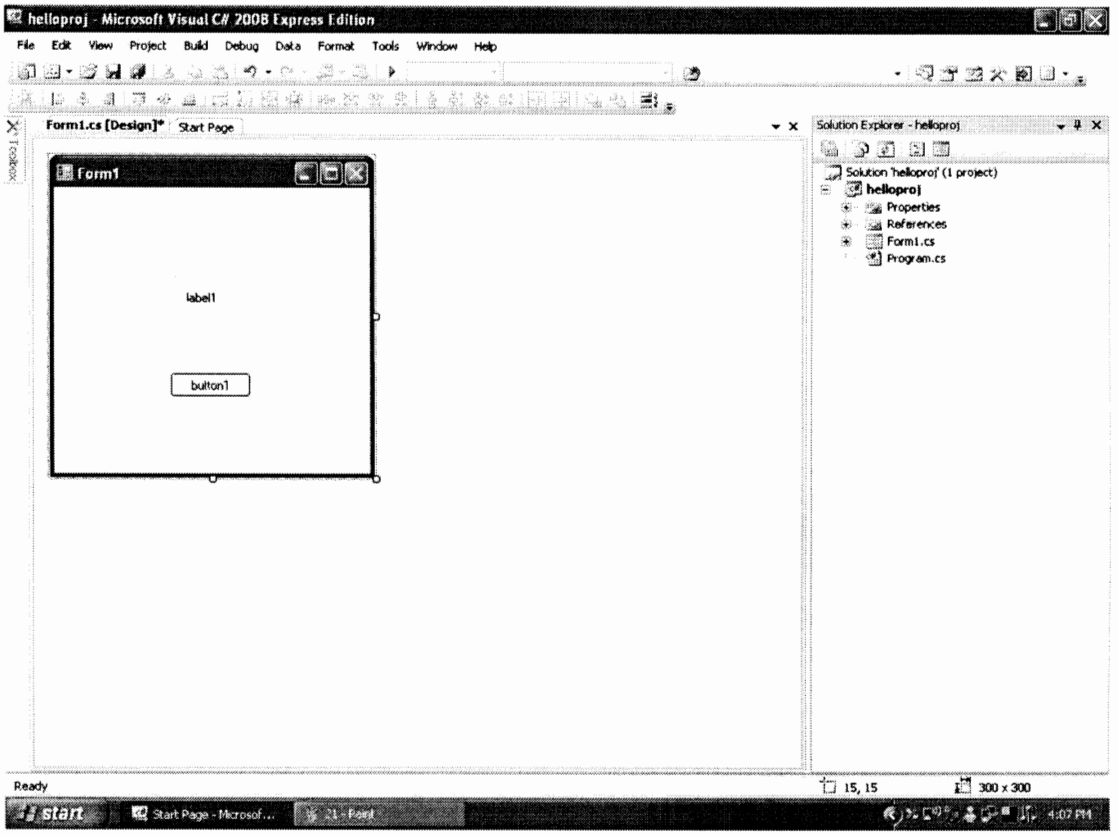

Figure 4.3. Label and Button components on the form

6. Double click the button component and type the following code.

```
//Changing the property of label1 component
label1.Text = "Hello World";
```

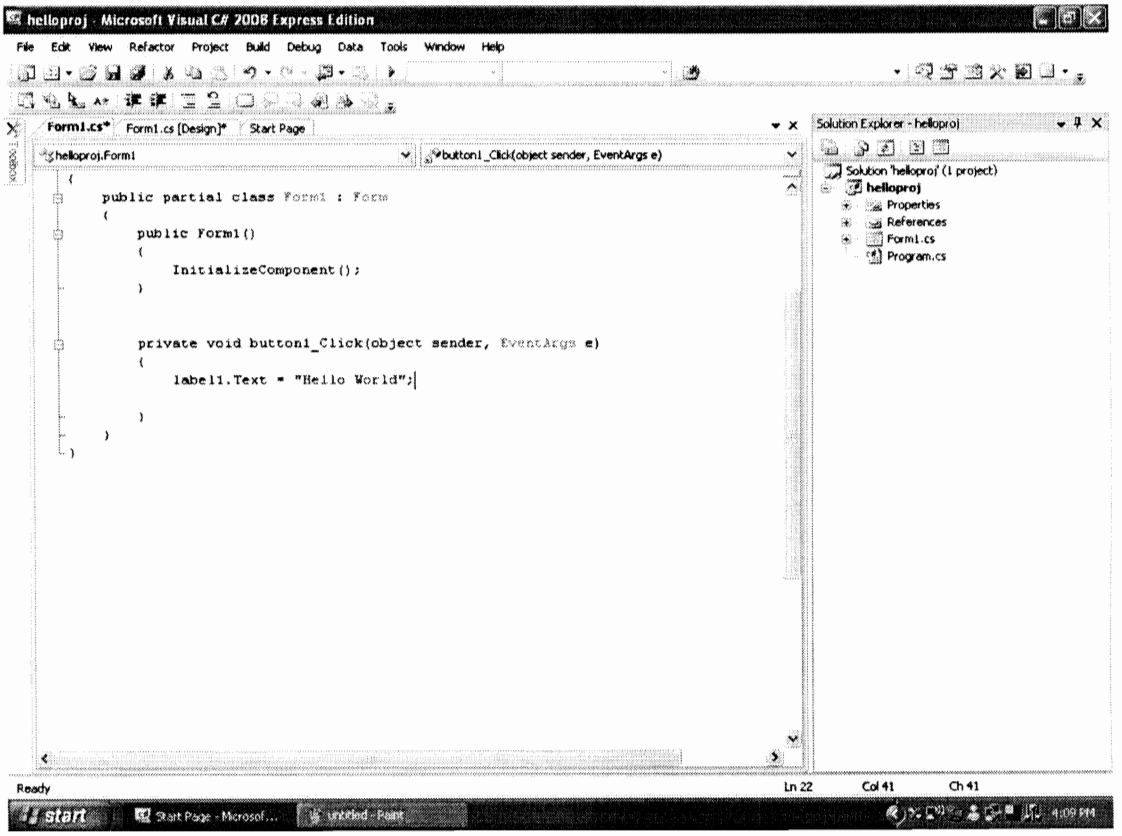

Figure 4.4. *Writing codes*

7. Save the project by clicking "File-->Save all" and providing a name as "helloproj".

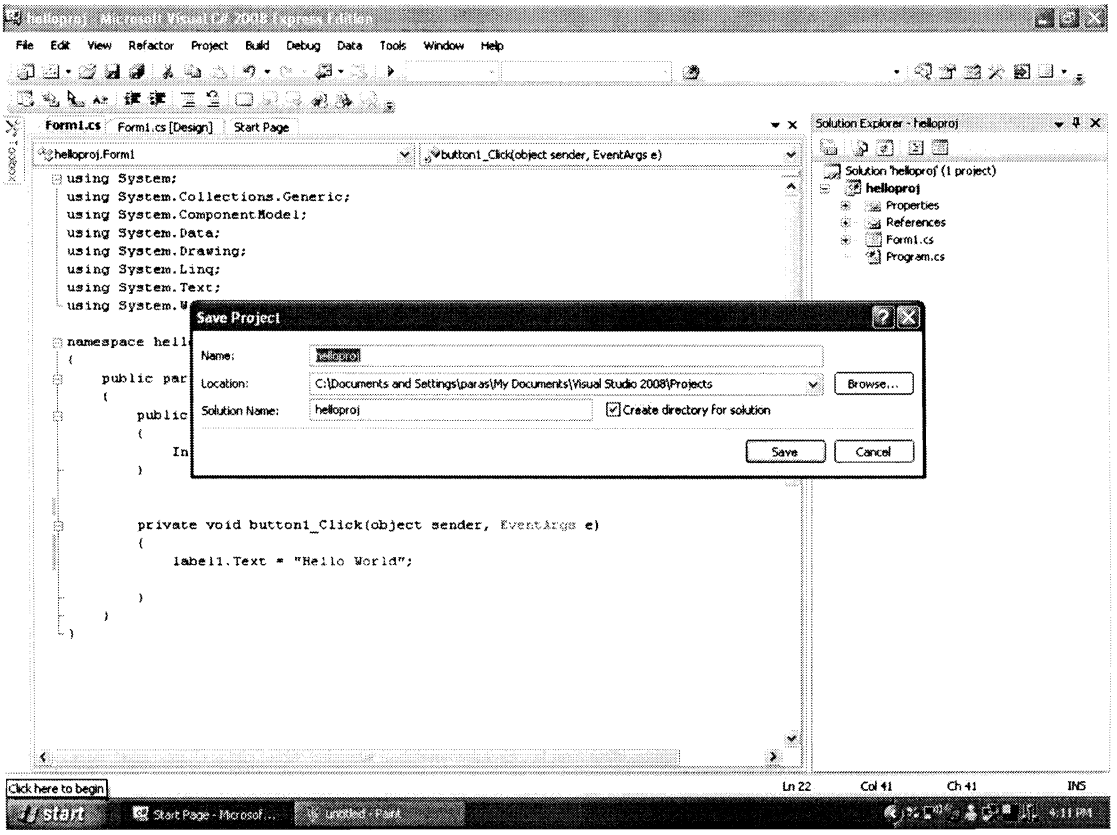

Figure 4.5. Saving project

8. Press CTRL + F5 to compile and run the application.

9. When you click the button, the label1 will be changed to "Hello World".

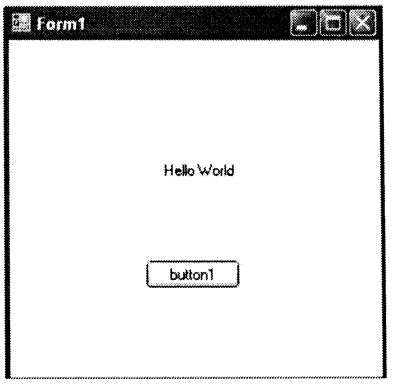

Figure 4.6. Sample application running

10. By default, the compiled executable file of the application will be placed in the folder C:\Documents and Setting\username\My Documents\Visual Studio 2008\Projects\helloproj\helloproj\bin\Release.

11. Properties of the components can be changed by right clicking the component and then clicking properties.

12. The files and references used in the project can be managed by clicking "View → Solution Explorer".

4.1.3 Downloading and Installing VeriLook SDK

A trial version of VeriLook SDK developed by Neurotechnologija can be downloaded from its official website http://www.neurotechnologija.com. The trial version runs for a period of 30 days. The full version of VeriLook SDK can be ordered online from the company's official site.

The installation procedures of the trial version of VeriLook were identified as:

1. Download the VeriLook SDK trial from the URL http://www.neurotechnologija.com/download.html#vl.

2. Unzip it and execute the file named verilookstadnardsdktrialsetup.

3. Simply click the next buttons with default settings on the upcoming windows.

4. After successful installation, the SDK needs to be activated.

5. Run the SDK Activation Wizard from Start → Neurotechnologija →VeriLook SDK Standard Trial → SDK (32-bit).

6. The activation wizard will popup and click Next.

7. Click "Install license manager service" to activate the SDK.

8. To verify if the SDK is activated or not, click the previous button and press the Diagnostics button.

9. The result must indicate "IsReg: Yes" as shown in the following screenshot.

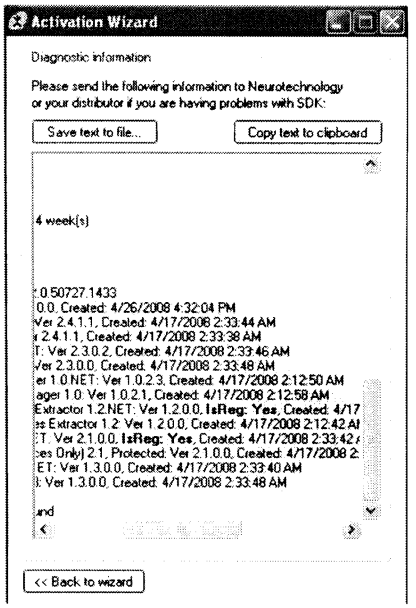

Figure 4.7. VeriLook Activation

If a license is purchased, the full version SDK download URL will be provided by the current vendor. The typical steps to install the full version of SDK are:

1. Download the SDK from the URL provided by VeriLook vendor.

2. Install the SDK by using the default settings.

3. Run the SDK Activation Wizard from Start-->Neurotechnologija-->VeriLook 3.0 SDK Standard --> SDK (32-bit).

4. During the time of activation, the hardware id of the system must be emailed to the company and the license file according to the hardware id is emailed back to the user that needs to be placed in the main folder of VeriLook SDK usually in C:\Program Files\Neurotechnologija\VeriLook 3.0 Standard SDK\bin\Win32_x86\Activation.

4.1.4 Configuring Microsoft Visual C# to Utilize VeriLook Libraries and Functions

SDK provides additional functionalities to the development environment. VeriLook adds functions and libraries for face recognition. In order to integrate the SDK functions and libraries, the related DLL files of VeriLook SDK must be properly referenced. The following methods were used to integrate VeriLook functions and libraries in the Microsoft Visual C# Express Edition environment.

1. Click Add Reference by right clicking the References in the solution explorer.
2. Select the Browse tab.
3. Open the location C:\Program Files\Neurotechnologija\VeriLook 3.0 Standard SDK\bin\Win32_x86\.
4. One of the five DLL files must be referenced in order to use their built-in functions. They are:

- Neurotech.dll

  This library references the NCore library of VeriLook SDK, which provides infrastructure for Neurotechnologija components.

- Neurotech.Images.dll

  It references the NImages library that is used to load, save, and convert images.

- Neurotech.Cameras.CameraMan.dll

  This library is the reference for CameraMan library and provides the functions for the management of the cameras.

- Neurotech.Biometrics.VLExtractor.dll

  This library is the reference to the VLExtractor library and is used to extract faces from images or frames for creating templates.

- Neurotech.Biometrics.VLMatcher.dll

This library is the reference to the VLMatcher library, which is used to match templates to declare match or non-match.

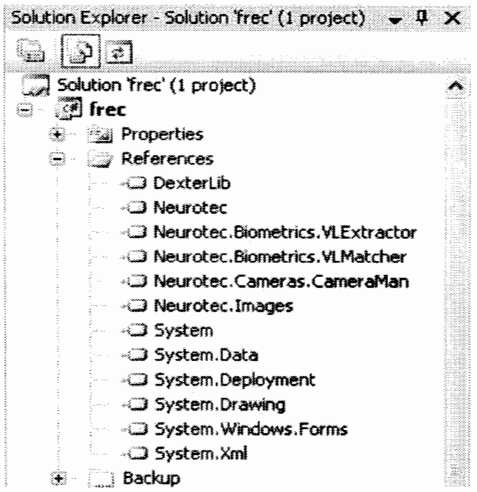

Figure 4.8. References of Veri-Look Libraries

5. Click the DLL files one at a time and press the OK button. This process enables the development environment to use the VeriLook libraries and functions.

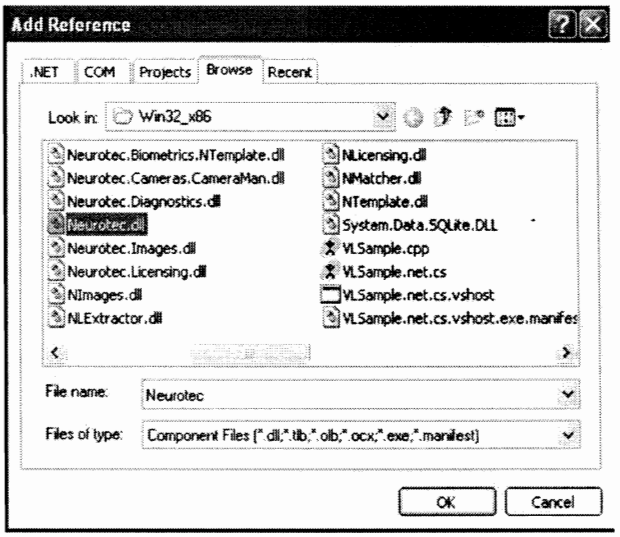

Figure 4.9. Adding references to VeriLook SDK

6. To utilize the referenced DLL files in the code segment, add the following references on the top section of every code view of C# files (i.e. *.cs files) which can be accessed using Solution Explorer.

```
using Neurotec.Biometrics;
using Neurotec.Images;
using Neurotec.Cameras;
```

4.2 FRECPROJ Prototype Description

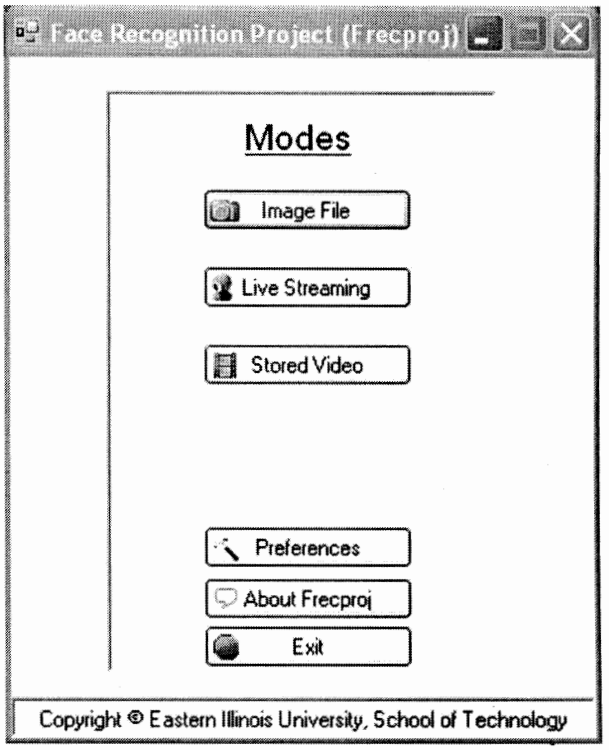

Figure 4.10. Project main from

FRECPRO is a software prototype, which was developed to study the development aspects of face recognition systems. The prototype was developed in order to facilitate understating on how to develop/integrate face recognition systems using VeriLook SDK.

FRECPROJ can enroll and match users based on still image files, live streaming from a web camera and recorded video files. It is also capable of enrolling multiple faces from images, live streaming and video files. The matching threshold value can be also set or changed from the user interface.

FRECPROJ can enroll multiple faces from various sources, but the matching sample must consist of a single face in image file mode or live streaming mode. In case of the video files, FRECPROJ can identify faces even if the video file has more than one face. FRECPROJ was programmed to work with images, live streaming and video files having a minimum resolution of 640x480 pixels.

FRECPROJ and its source code and executable file are available for downloading without any charge from the URLs http://pen.eiu.edu/~ppradhan/frecproj/ and http://www.eiu.edu/~pingliu/frecproj/. It is fully authorized to make changes, redistribute and make copies of the software for research and study purposes.

4.3 FRECPROJ Main Components

The prototype developed in this research involves enrolling and matching using still image files, web camera or video files such as AVI and WMV. The three (3) modules of FRECPROJ are explained below with sample codes written in Visual C#.NET.

4.3.1 Enrolling and Matching Using Still Images

This section illustrates basic procedures on enrolling and matching of face(s) using still JPEG images. It may enroll a single face or multiple faces from an image.

4.3.1.1 Enrollment (imagemodeverificationenroll.cs, imagemodeidentificationenroll.cs)

1. In order to enroll a still image, the image file has to be open and input. The enrollment process starts with opening the image file by using the OpenFileDialog component of Visual C#. Users will be able to browse to the directory and select the image file.

```
// Initializing the File dialog box component
OpenFileDialog enrollfile = new OpenFileDialog();

// Filter the file dialog box for JPEG file only
enrollfile.Filter = "JPEG Files|*.jpg;*.JPG";

// Displays the image selection window
enrollfile.ShowDialog();
```

2. The image file is checked if it is in JPEG format and has the minimum resolution of 640x480.

```
// Declaring the variable as Image type
public Image eimage;

// Assigning the image file to eimage image type variable
eimage = Image.FromFile(enrollfile.FileName);

// Extract the extension of the image file
string extension = Path.GetExtension(enrollfile.FileName);

// Converting the extension of the image file to lower case
string extension_lower = extension.ToLower();

// Previews the image file in the picture box component if the image is of JPEG type and its minimum
resolution of 640x480
if (((eimage.Width >= 640) & (eimage.Height >= 480)) & ((extension_lower == ".jpg") | (extension_lower ==
".jpeg")))

{
        enrolltextbox.Text = enrollfile.FileName;
        enrollpicturebox.Image = Image.FromFile(enrollfile.FileName);

}
else
{
        MessageBox.Show("Image requirement failed");
}
```

In order to check the image validation, an Image type variable named "eimage" provided by Visual C# is declared. Then, the image is loaded to the "eimage" variable

and to check for the proper width and height by accessing the width and height properties of "eimage". To check the valid extension of the image file, function such "Path.GetExtension" is used.

3. The image is converted to grayscale.

```
// Loading the image file
NImage image = NImage.FromFile(enrolltextbox.Text);

// Converting the image to gray scale, output in this case is grayscale
NGrayscaleImage grayscale = (NGrayscaleImage)NImage.FromImage(NPixelFormat.Grayscale, 0, image);
```

4. The extractor is initialized.

```
// Initialization of VeriLook extractor
VLExtractor extractor = new VLExtractor();
```

5. A template is created from an image consisting of a single face.

```
// Declearing a structure which is used to hold face detection results such as location of eyes, face(s)
VleDetectionDetails details;

// Creating a template variable and assigning face detection results in a structure
byte[] template = extractor.Extract(grayscale, out details);
```

6. The template is compressed and stored if the template has a detected face.

```
// Compressing and storing template if a face is found.
if ((details.FaceAvailable) & (details.EyesAvailable))
{
        // Compressing the template using compress method of VeriLook extraetor
        byte[] compressed = extractor.Compress(template);

        // Using file input/output to store the template
        // Creating a file named enroll.dat
        FileStream fs = File.Create("c:\\frecproj\\frecdata\\templates_db\\temporary\\enroll.dat");

        // Initialization of a binary writer to work with binary files
        BinaryWriter writer = new BinaryWriter(fs);

        // Writing the template to enroll.dat file including the template length
        writer.Write(compressed.Length);
        writer.Write(compressed);

        // Closing the binary writer and file stream
        writer.Close();
        fs.Close();
}
```

Before storing a template, it is compressed using a SDK compress method "extractor.Compress". In order to store the template, file input output functions of Visual C# is used. In this case a binarywriter is initiated that points to the destination filename and is used to save template in form of compressed binary files using "Write" method.

7. A rectangle is drawn around the detected face in an image.

```
// Drawing rectangle and adjusting for preview in 640x480 in case of image size greater than 640x480

// Creating a drawing layer over the picturebox component and initialization of a drawing pen
Graphics g = enrollpicturebox.CreateGraphics();
Pen pen = new Pen(Color.Blue);

// Finding and Setting the X, Y co-ordinates, height and width of the detected face
double rx = details.Face.Rectangle.X + .00;
double ry = details.Face.Rectangle.Y + .00;
double rw = details.Face.Rectangle.Width + .00;
double rh = details.Face.Rectangle.Height + .00;

// New X,Y, Width and Height are calculated in order to normalize for images greater than 640x480
double newrx = (rx / eimage.Width) * 640;
double newry = (ry / eimage.Height) * 480;
double newrw = (rw / eimage.Width) * 640;
double newrh = (rh / eimage.Height) * 480;

// Initializing the rectangle in the form of Rectangle(X,Y,Width,Height)
Rectangle rect = new Rectangle(Convert.ToInt32(newrx), Convert.ToInt32(newry), Convert.ToInt32(newrw),
Convert.ToInt32(newrh));

// Drawing the rectangle
g.DrawRectangle(pen, rect);
```

A rectangle is drawn using built-in graphics function of Visual C#. First a graphical layer is added to the preview picture box, then a drawing pen is initialized and finally the DrawRectangle method is invoked to draw a rectangle using the coordinates, height and width of the face which are provided by VeriLook extractor.

8. Enrollment from an image consisting of one or multiple faces. Steps 1 to 4 would be the same.

```
// Initializing an array structure VleFace[] to detect face(s) to find information of location of face(s)
VleFace[] faces = extractor.DetectFaces(grayscale);
// Looping through all the detected faces. In this case the face template(s) is/are generated with the help of
```

```
found eyes or eyes detection
for (int i = 0; i < faces.Length; ++i)
{
        //Initialization of VleEyes[] array structure to detect eyes in a face
        VleEyes eyes = extractor.DetectEyes(grayscale, faces[i]);

        // Create template variable with the help of eyes in the found face
        byte[] template = extractor.Extract(grayscale, eyes);

}
```

In this case, an array named "faces" is declared, which has the information of detected faces by "extractor.DetectFaces" SDK function. The extractor detects the total number of faces and a C# for loop is used to iterate through all the detected faces to create templates. To generate face templates, face detection using eyes is used in which the information of eyes are used to differentiate faces in creating templates.

9. The template(s) are compressed and stored. Before the template(s) are stored, the program will check if the template(s) already enrolled.

```
// Checking templates if already enrolled
// Performing identification process with the stored templates

// Declaring and initializing a VeriLook Matcher with the current template
VLMatcher matcher = new VLMatcher();
matcher.IdentifyStart(template);

// Counting previously enrolled stored template file(s)
string[] files = Directory.GetFiles("c:\\frecproj\\frecdata\\templates_db\\permanent\\");
int count = files.Length;

// Reading all stored templates and storing in an jagged type array
// Declaration of arrays storedtemplates[][] and sizes[]
byte[][] storedtemplates = new byte[count][];
int[] sizes = new int[count];

// Looping through all the stored template file(s) and assigning the template(s) to storedtemplates array
for (int j = 0; j < count; ++j)
{
        // Creating a file stream that points to stored template
        FileStream chfs = File.OpenRead(files[j]);

        // Initialization of a binary reader
        BinaryReader reader = new BinaryReader(chfs);

        // Assigning the size of the template file to array sizes[] by reading only the integers of the
template file
        sizes[j] = reader.ReadInt32();

        // Based on the template size create storage space in array and read the content of template file
```

```
into the array
            storedtemplates[j] = new byte[sizes[j]];

            reader.Read(storedtemplates[j], 0, sizes[j]);

            // Closing the binary reader.
            reader.Close();
}

// Using VeriLook matcher to check if previously enrolled

double score;
maxscore = 0;

// Looping through all the array elements of storedtemplates
for (int k = 0; k < count; ++k)
{

            // Generating a score using matcher functions and current template
            score = matcher.IdentifyNext(storedtemplates[k]);

            // If the score generated is greater than the threshold value 0.65 , then it is resulted that the
template has been already enrolled previously
            if (score > 0.65)
            {
               maxscore = score;
               MessageBox.Show("Face already in the database");
            }
            score = 0;
}

// Checking ends here

// Enrolls the template if only previously not enrolled.
if (maxscore < 0.65)
{
            // Compressing the template
            byte[] compressed = extractor.Compress(template);

            // Template naming using the random variable
            Random rand = new Random();
            int randomnumber = rand.Next();
            x =  Convert.ToString(randomnumber) + ".tem";

            // Creating a filestream to store template using the variable x
            FileStream fs = File.Create("c:\\frecproj\\frecdata\\templates_db\\permanent\\" + x);


            // Initialization of a binary writer and stoing the compressed template
            BinaryWriter writer = new BinaryWriter(fs);
            writer.Write(compressed.Length);
            writer.Write(compressed);

            // Closing binary writer and file stream
            writer.Close();
            fs.Close();
}
```

Before storing the templates using the File Input Output functions of Visual C#, they are checked against the stored templates. A matcher is initiated and initialized with a template. The stored templates are also loaded into a separate array using Visual C# binaryreader object. Then, the template is compared against the items of an array using the matching threshold. If the score generated less than the matching threshold value, it is stored using Visual C# binarywriter, otherwise ignored.

10. Rectangle(s) around the detected face(s) in an image is/are drawn

```
// Draw rectangle for image size of 640x480 and greater
// Variable i is used from step 8 to differentiate face(s)

// Creating a drawing layer over the picturebox component and initialization of a drawing pen
Graphics g = enrollpicturebox.CreateGraphics();
Pen pen = new Pen(Color.Blue);

// Finding and Setting the X, Y co-ordinates, height and width of the detected face
double rx = faces[i].Rectangle.X + .00;
double ry = faces[i].Rectangle.Y + .00;
double rw = faces[i].Rectangle.Width + .00;
double rh = faces[i].Rectangle.Height + .00;

// New X,Y, Width and Height are calculated in order to normalize for images greater than 640x480
double newrx = (rx / eimage.Width) * 640;
double newry = (ry / eimage.Height) * 480;
double newrw = (rw / eimage.Width) * 640;
double newrh = (rh / eimage.Height) * 480;

// Initializing the rectangle in the form of Rectangle(X,Y,Width,Height)
Rectangle rect = new Rectangle(Convert.ToInt32(newrx), Convert.ToInt32(newry), Convert.ToInt32(newrw),
Convert.ToInt32(newrh));

// Drawing the rectangle
g.DrawRectangle(pen, rect);

// Displaying face number inside the rectangle using DrawString function of C#
x = "face" + (int)(i+1);
g.DrawString(x, new Font("verdana", 10), new SolidBrush(Color.Blue), Convert.ToInt32(newrx),
Convert.ToInt32(newry));
```

Rectangles are drawn around the detected face(s) using DrawRectangle method of Visual C#. Also the title for the face such as face0 and face1 is also drawn using DrawString method.

4.3.1.2 Matching (imagemodeverificationmatch.cs, imagemodeidentificationmatch.cs)

1. The matching process starts with opening an image file by using OpenFileDialog component. Users will be able to browse to the directory and select the image file.

```
// Initializing the File dialog box component
OpenFileDialog matchfile = new OpenFileDialog();
// Filter the file dialog box for JPEG file
matchfile.Filter = "JPEG Files|*.jpg;*.JPG";
// Pops up the image selection window
matchfile.ShowDialog();
```

2. The minimum requirements of the image file for JPEG type and resolution of 640x480 are checked.

```
// Declaring the variable as Image type
public Image mimage;

// Assigning the image file to mimage image type variable
mimage = Image.FromFile(matchfile.FileName);

// Extracting the extension of the image file
string extension = Path.GetExtension(matchfile.FileName);

// Converting the extension of the image file to lower case
string extension_lower = extension.ToLower();

// Previews the image file in the picture box component if the image is of JPEG type and its minimum
resolution of 640x480
if (((mimage.Width >= 640) & (mimage.Height >= 480)) & ((extension_lower == ".jpg") | (extension_lower ==
".jpeg")))

{
        matchtextbox.Text = matchfile.FileName;
        matchpicturebox.Image = Image.FromFile(matchfile.FileName);

}

else

{
        MessageBox.Show("Image requirement failed");
}
```

3. A template is created from an image. Note that the matching image should not consist of more than one face.

```
// Creating matching template which will be used for verification and identification purposes
// Loading the image file
NImage image = NImage.FromFile(matchfile.FileName);

// Converting the image to gray scale
NGrayscaleImage grayscale = (NGrayscaleImage)NImage.FromImage(NPixelFormat.Grayscale, 0, image);

// Extractor initialization
VLExtractor extractor = new VLExtractor();

// Declaring a structure which is used to hold face detection results
VleDetectionDetails details;

// Creating a matching template
byte[] template = extractor.Extract(grayscale, out details);
```

4. Verification.

In this case, the matching template is compared against the enrolled template. The process is 1:1 comparing. The process reads the enrolled template and the VeriLook matcher compares this template with the matching template to generate a score and decide a match or no match.

4.1 Enrolled template is read into an array.

```
//Reading enrolled template

// Initialization of a file stream to read enrolled template
FileStream fs = File.OpenRead("c:\\frecproj\\frecdata\\templates_db\\temporary\\enroll.dat");

// Creating binary reader object
BinaryReader reader = new BinaryReader(fs);

// Assigning the size of the enrolled template file to array sizes[] by reading only the integers
int size = reader.ReadInt32();

// Based on the template size create storage space to read the content of template file
byte[] enrolledtemplate = new byte[size];

// Reading the template and assigning it to enrolledtemplate
reader.Read(enrolledtemplate, 0, size);

// Closing the binary reader
reader.Close();
```

Using the C# binaryreader, the enrolled template is stored into a variable. This variable is compared with the template created in step 3.

4.2 A matcher is used to declare a match or no match result.

```
// Initialization of a matcher
VLMatcher matcher = new VLMatcher();

// Generating score and declaration of match or no match using matcher.verify method considering the
threshold value of 0.65
double score = matcher.Verify(template, enrolledtemplate);
if (score > 0.65)
{
                MessageBox.Show("Percentage matched : " + Convert.ToString(Math.Round(score * 100, 1)) +
"%");
 }

else
{
                MessageBox.Show("Match failed");
}
```

In order to decide a match or no match, matcher is initiated and using the SDK function "matcher.Verify", the score is generated which is used to declare a match or no match.

5. Identification.

This is the 1:M matching process in which a matching template is compared against the previously enrolled template(s). If the matching score is greater than the threshold value, then it is declared as a match.

5.1 A matcher is initialized with the matching reference template.

```
// Matcher initialization
VLMatcher matcher = new VLMatcher();

// Using template as a reference template
 matcher.IdentifyStart(template);
```

5.2. Enrolled templates are read into an array using C# binaryreader.

```
// Counting stored template files
string[] files = Directory.GetFiles("c:\\frecproj\\frecdata\\templates_db\\permanent\\");
int count = files.Length;
// Reading all stored templates and storing in an jagged type array
```

```
// Declaration of an arrays storedtemplates and sizes
byte[][] storedtemplates = new byte[count][];
int[] sizes = new int[count];

// Looping through all the stored template file(s) and assigning the template(s) to storedtemplates array
for (int i = 0; i < count; ++i)
{
                // Creating a file stream that points to stored template
                 FileStream fs = File.OpenRead(files[i]);

                // Initialization of a binary reader
                BinaryReader reader = new BinaryReader(fs);

// Assigning the size of the template file to array sizes[] by reading only the integers of the template file
                sizes[i] = reader.ReadInt32();

                 // Based on the template size create storage space in array and read the content of template
file into the array
                 storedtemplates[i] = new byte[sizes[i]];
                 reader.Read(storedtemplates[i], 0, sizes[i]);

                // Closing the binary reader.
                reader.Close();

}
```

5.3. The matching process uses the VeriLook matcher to declare match or no match.

```
double maxscore = 0;

// Declaration of a scores array to store series of generated scores
double[] scores = new double[count];

// Looping through all the array elements of storedtemplates
for (int i = 0; i < count; ++i)
{

                 // Generating score and assigning them to array
                 double score = matcher.IdentifyNext(storedtemplates[i]);
                scores[i] = score;

                 // Stores the maximum value of score generated in the whole loop process
                 if (score > maxscore)

                 {
                   maxscore = score;
                 }
                score = 0;
}

// Declaration of match or non match based on variable maxscore
if (maxscore > 0.65)
{

                MessageBox.Show("Percentage matched : " + Convert.ToString(Math.Round(maxscore * 100, 1))
+ "%");
}
 else
{
```

```
            MessageBox.Show("Match failed");
}
```

In this case, using the C# for loop and "matcher.IdentifyNext" SDK function, the template is compared against all the stored templates. Matching scores are generated in the process. If any one among the generated scores is greater than the threshold value, then it is declared as match otherwise a no match.

4.3.2 Enrolling and Matching Using Live Streaming Mode by Web Camera

In order to enroll and match face(s) using a web camera, at first the video control component was created using panel object and drawing library. The video control component was created to enroll and match face(s). The following codes illustrate the basic steps used on creating a video control component.

4.3.2.1 Creating Video Control Component

1. A panel component from the Toolbox is added in a form window.
2. By double clicking the form, the name of the form object is changed to

```
public class VideoControl : System.Windows.Forms.Panel
```

3. Then the three properties for the video control component are created.

- image property: This property is used to pass the image frame from the program to the video control. The get and set method are used to read and write image frame, a typical practice in C#.

```
private Bitmap image;
public Bitmap Image
{
```

```
        get
        {
          return image;
        }

        set
        {
          image = value;
        }
}
```

- faces property: This property is used to read and write the coordinates of detected face(s).

```
private VleFace[] faces;
 public VleFace[] Faces
 {
        get
        {
          return faces;
        }

        set
        {
          faces = value;
        }
 }
```

- detectiondetails property: This is used to flush or invalidate the existing frame which is being displayed to draw or display the next image frame.

```
private VleDetectionDetails detectionDetails;
public VleDetectionDetails DetectionDetails
{
        get
        {
          return detectionDetails;
        }

        set
        {
          Invalidate();
        }
}
```

4. Under the OnPaint event of the video control component (that can be accessed using events menu of the property windows of the component), the following codes are used to display a frame that is passed from the program.

```
// Creating a graphics object
Graphics g = pe.Graphics;

// Draws only the image passed from the main program
if (image != null)
{
        // Draws the image passed from the program
        g.DrawImage(image, 0, 0, 640, 480);
}
```

In the above code, the DrawImage method provided by graphics library of Visual C# displays the image frame at location X, Y in the resolution of 640 by 480.

5. To draw the rectangle(s) around the detected face(s) , the following codes are used under the OnPaint event of the video control component.

```
// Draws rectangle using pen and drawrectangle method
// Checks if face(s) is/are available
if ((faces != null) && (faces.Length != 0))
{
        Pen pen = new Pen(Color.Blue, 3);

        // ivmode is passed from the program to detect if the mode is 1:1 or 1:M. If 1:1, there will be only
one face and only one rectangle is drawn. If the mode is 1:M, there can more than one face(s) and
rectangle(s)
        if (ivmode == "ver")
        {
          g.DrawRectangle(pen, faces[0].Rectangle);
        }

        if (ivmode == "idn")
        {
          // Draws rectangles around every detected face(s)
          for (int i = 0; i < faces.GetLength(0); ++i)
          {
            g.DrawRectangle(pen, faces[i].Rectangle);
          }

}
```

In the verification mode, the live streaming consists of single face. Hence, if the mode is verification, a single rectangle is drawn around a single detected face. If the mode is identification it may consists of one or more faces. Thus, in this case one or more rectangles are drawn around the detected face(s). "faces.GetLength" function is used to find total number to faces in live streaming and DrawRectangle method is used to draw rectangles.

6. The following codes are used to remove the background from being painted.

```
// This is necessary to prevent the background from being painted. If this code is not use the continuous
black strips are seen during the video streaming

protected override void OnPaintBackground(PaintEventArgs pevent)
{
        // do nothing
}
```

Note: Complete code for video control can be seen in Appendix (myvideo.cs)

4.3.2.2 Using of Video Control Component in the Form

After creating the video control component, the component appeared in the Toolbox of Visual C#. The video control component was dragged into the form. To display the live streaming from a web camera, the form_Load event of the form was coded in a way that when the form is loaded the video control component of the form displays the live streaming captured by the web camera. The following illustrates the process to display live streaming on to the form.

1. By double clicking somewhere in the form, the form load event of the form is opened.

2. The camera is initialized and a thread is created to run the streaming process in the background.

```
// Initialization of connected camera
// Detects the connected camera and assign it as activeDevice. From this onward activeDevice is referred by
connected web camera
cameraMan = new CameraMan(this);
private CameraMan cameraMan;
private Camera activeDevice;
foreach (Camera camera in cameraMan.Cameras)
{
        activeDevice = camera;
}

// Creating a thread to run in background continuously
// Calls the function named GetFrames()
private Thread getFrameThread = null;
getFrameThread = new Thread(new ThreadStart(GetFrames));
getFrameThread.Start();
```

The camera object is initialized as cameraMan using SDK function, which provides information on connected web camera. The connected camera is accessed by the method "cameraMan.Cameras". After the camera has been initialized, it is referred by activeDevice. A thread getFrameThread is initialized which runs continually on the background and invoking the GetFrames() function.

3. The GetFrames() function is created which starts the connected camera and captures the current frame from the web camera.

```
private bool working = true;
private void GetFrames()
{
        working = true;

        // Starting the camera
        activeDevice.StartCapturing();

        // Capture the current frame
        while (working)
        {
           NImage image = activeDevice.GetCurrentFrame();

           // Calls the Delegate SetCurrentFrame()
           SetCurrentFrame(image);
        }
}
```

In order to start the camera, "activeDevice.StartCapturing" is used. After the camera has been started "activeDevice.GetCurrentFrame()" method is called to capture the current frame that is passed to function SetCurrentFrame().

4. "A delegate is an object that is created to refer to a static method or an instance method and then used to call this method (aspfree.com, C# Delegates Explained, ¶ 2). The delegate named SetCurrentFrame is created.

```
private delegate void GetFrameEventHandler(NImage image);
private void SetCurrentFrame(NImage image)
{
        //Runs for the first time to create a delegate using image
        if (this.InvokeRequired)
        {
          GetFrameEventHandler getFrameEventHandler = new GetFrameEventHandler(SetCurrentFrame);
          this.Invoke(getFrameEventHandler, new object[] { image });
        }
        // then runs this and calls the functions SetCurrentImage after the delegate is created
        else
        {
          SetCurrentImage(image);
        }
}
```

5. The image value or frame is passed to the video control using SetCurrentImage() function. Along with the current image value, the function also sets/gets coordinates of the detected faces and invalidate/dispose the current image frame to assist the display to next upcoming frame.

```
extractor = new VLExtractor();
private VleDetectionDetails detectionDetails;
private VleFace[] faces;
private VLExtractor extractor;
private void SetCurrentImage(NImage image)
{

        // Converting image into grayscale
        NGrayscaleImage grayImage = (NGrayscaleImage)NImage.FromImage(NPixelFormat.Grayscale, 0,
image);

        // Converting image to Bitmap type before drawing
        Bitmap bm = image.ToBitmap();

        // Defining the location and size of the video control to show in the form
```

```
        videoControl1.SetBounds(8, 25, (int)image.Width, (int)image.Height);

        // Passing the current image frame to videocontrol object to display in the form
        videoControl1.Image = new Bitmap(bm);

        // To invalidate a frame
        videoControl1.DetectionDetails = detectionDetails;
        // Pass ivmode = "ver" in Verification mode and ivmode= "idn" in Identification mode
        videoControl1.ivmode = "ver"; // For verification mode, use this code
        videoControl1.ivmode = "idn"; // For identification mode, use this code

        // Detects faces from grayscale image
        faces = extractor.DetectFaces(grayImage);

        // Necessary to draw rectangles in face(s)
        videoControl1.Faces = faces;

        // Disposing objects
        image.Dispose();
        grayImage.Dispose();
        bm.Dispose();
}
```

The captured image from the web camera is passed to the video component using "videocontrol1.Image = new Bitmap(bm)" where, bm is the image converted to Bitmap format. In addition, the face detection details, verification or identification mode is passed to the main video control component.

6. The following codes are used to stop the camera when the form containing the video control component is closed.

```
private void livemodeverificationenroll_FormClosing(object sender, FormClosingEventArgs e)
{
        // Stop the thread and camera
        if (getFrameThread != null && getFrameThread.IsAlive)
        {
            working = false;
            while (getFrameThread.IsAlive)
            {
                Application.DoEvents();
            }
        }
        activeDevice.StopCapturing();

        activeDevice = null;
}
```

To stop the running camera, the method "activeDevice.StopCapturing()" is used.

4.3.2.3 Enrollment (livemodeverificationenroll.cs, livemodeidentificationenroll.cs)

1. The enrollment process from a live streaming web camera uses the video control component created as shown in the section 4.3.2.1.

2. The Form load event of the enrollment form must be equipped using the codes as shown in the above section 4.3.2.2.

3. When the live streaming is running, the desired frame is captured and saved.

```
// Captures and save the current frame or image
activeDevice.GetCurrentFrame().Save(@"c:\\frecproj\\frecdata\temp_images\enroll.jpg");
```

4. After the image is saved, the saved image is loaded and converted to gray scale.

```
// Loading the image file
NImage image = NImage.FromFile(@"c:\\frecproj\\frecdata\temp_images\enroll.jpg");

// Converting the image to gray scale
NGrayscaleImage grayscale = (NGrayscaleImage)NImage.FromImage(NPixelFormat.Grayscale, 0, image);
```

5. The extractor is initialized.

```
//Initialization of VeriLook extractor
VLExtractor extractor = new VLExtractor();
```

6. A template is extracted from a captured image consisting of a single face.

```
// Declaring a structure which is used to hold face detection results
VleDetectionDetails details;

// Creating a template and assigning face detection results in a structure
byte[] template = extractor.Extract(grayscale, out details);
```

7. The template is compressed and stored if a face is identified and the template saved.

```
// Compressing and storing template if a face is found.
if ((details.FaceAvailable) & (details.EyesAvailable))
{
        // Compressing the template using compress method of VeriLook extractor
```

```
            byte[] compressed = extractor.Compress(template);

            // Using file input/output to store the template
            // Creating a file named enroll.dat
            FileStream fs = File.Create("c:\\frecproj\\frecdata\\templates_db\\temporary\\enroll.dat");

            // Initialization of a binary writer to work with binary files
            BinaryWriter writer = new BinaryWriter(fs);

            // Writing the template to enroll.dat file with the template length
            writer.Write(compressed.Length);
            writer.Write(compressed);

            // Closing the binary writer and file stream
            writer.Close();
            fs.Close();
}
```

To check if the faces are available, "details.FaceAvailable" and "details.EyesAvailable" provided by detection details are used.

8. To enroll from a saved image consisting of one or multiple faces, templates are created with the help of eyes detection using SDK function "VleEyes."

```
// Initializing an array structure VleFace[] to detect face(s) to find information of location of face(s)
VleFace[] faces = extractor.DetectFaces(grayscale);

// Looping through all the detected faces. In this case the face template(s) is/are generated with the help of
eyes detection
for (int i = 0; i < faces.Length; ++i)
        {
          // Initialization of VleEyes[] array structure to detect eyes in a face
          VleEyes eyes = extractor.DetectEyes(grayscale, faces[i]);
          // Create template variable with the help of eyes in the detected face
          byte[] template = extractor.Extract(grayscale, eyes);
        }
```

9. Template(s) are compressed and stored. Before storing the template(s), program checks if the template(s) is/are already enrolled.

```
// Checking if already enrolled

// Performing identification process with the stored templates
// Declaring and initializing a VeriLook Matcher with the current template
VLMatcher matcher = new VLMatcher();
matcher.IdentifyStart(template);
//Counting previously enrolled stored template file(s)
string[] files = Directory.GetFiles("c:\\frecproj\\frecdata\\templates_db\\permanent\\");
int count = files.Length;
```

```
// Reading all stored templates and storing in an jagged type array
// Declaration of  arrays storedtemplates and sizes
byte[][] storedtemplates = new byte[count][];
 int[] sizes = new int[count];

// Looping through all the stored template file(s) and assigning the template(s) to storedtemplates array
for (int j = 0; j < count; ++j)
{
              // Creating a file stream that points to stored template
               FileStream chfs = File.OpenRead(files[j]);

               // Initialization of a binary reader
              BinaryReader reader = new BinaryReader(chfs);

                // Assigning the size of the template file to array sizes[] by reading only the integers of the
template file
              sizes[j] = reader.ReadInt32();

                // Based on the template size create storage space in array and read the content of template
file into the array
               storedtemplates[j] = new byte[sizes[j]];
               reader.Read(storedtemplates[j], 0, sizes[j]);

               // Closing the binary reader.
              reader.Close();
}

//Using VeriLook matcher to check if previously enrolled
double score;
maxscore = 0;

//Looping through all the array elements of storedtemplates
for (int k = 0; k < count; ++k)

{
              // Generating a score value using matcher functions and current template
               score = matcher.IdentifyNext(storedtemplates[k]);

              // If the score generated in greater than the threshold value 0.65 , then it is resulted that the
template has been already enrolled previously
              if (score > 0.65)
              {
                 maxscore = score;
                 MessageBox.Show("Face already in the database");
              }

              score = 0;
}

// Checking ends here

// Enrolls the template if only previously not enrolled.
if (maxscore < 0.65)
{
              // Compressing the template
               byte[] compressed = extractor.Compress(template);

               // Template naming using the random variable
                Random rand = new Random();
                int randomnumber = rand.Next();
                x =  Convert.ToString(randomnumber) + ".tem";
               // Creating a filestream to store template using the variable x
```

```
            FileStream fs = File.Create("c:\\frecproj\\frecdata\\templates_db\\permanent\\" + x);
            // Initialization of a binary writer and storing the compressed template
            BinaryWriter writer = new BinaryWriter(fs);
            writer.Write(compressed.Length);
            writer.Write(compressed);

            // Closing binary writer and file stream
            writer.Close();
            fs.Close();
}
```

In order to check if the templates are already in the pool, SDK matcher is used.

4.3.2.4 Matching (livemodeverificationmatch.cs, livemodeidentificationmatch.cs)

1. The matching process from a live streaming web camera uses the video control component created as shown in the section 4.3.2.1.

2. The Form load event of the verification or identification form must be equipped using the codes as shown in the above section 4.3.2.2.

3. When the live streaming is running, the desired frame is captured and saved as a temporary JPG file.

```
// Capture the current frame and create template
activeDevice.GetCurrentFrame().Save(@"c:\\frecproj\\frecdata\\temp_images\\match.jpg");
```

4. A template is created from an image. Note that the matching image should not consist of more than one face.

```
// Creating matching template which will be used for verification and identification purposes
// Loading the image file
NImage image = NImage.FromFile(@"c:\\frecproj\\frecdata\temp_images\match.jpg");

// Converting the image to gray scale
NGrayscaleImage grayscale = (NGrayscaleImage)NImage.FromImage(NPixelFormat.Grayscale, 0, image);

// Extractor initialization
VLExtractor extractor = new VLExtractor();
// Declaring a structure which is used to hold face detection results
VleDetectionDetails details;
// Creating a matching template
byte[] template = extractor.Extract(grayscale, out details);
```

5. Verification.

During verification, the matching template is compared with the enrolled template. The process that implements is 1:1 comparison. The process reads the enrolled template and the VeriLook matcher compares this template with the matching template to generate a score and deciding a match or no match. VLMatcher, the matching function from the SDK, is used for this purpose.

5.1. The enrolled template is read into an array.

```
//Reading enrolled template

// Initialization of a file stream to read enrolled template
FileStream fs = File.OpenRead("c:\\frecproj\\frecdata\\templates_db\\temporary\\enroll.dat");
// Creating binary reader object
BinaryReader reader = new BinaryReader(fs);
// Assigning the size of the enrolled template file to array sizes[] by reading only the integers
int size = reader.ReadInt32();

// Based on the template size create storage space to read the content of template file
byte[] enrolledtemplate = new byte[size];

// Reading the template and assigning it to enrolledtemplate
reader.Read(enrolledtemplate, 0, size);

// Closing the binary reader
 reader.Close();
```

5.2 VeriLook matcher is initialized and used to declare a match or no match.

```
// Initialization of a matcher
VLMatcher matcher = new VLMatcher();

// Generating score and declaration of match or no match using matcher.verify method considering the
threshold value of 0.65
double score = matcher.Verify(template, enrolledtemplate);
if (score > 0.65)
{

                MessageBox.Show("Percentage matched : " + Convert.ToString(Math.Round(score * 100, 1)) +
"%");
 }
else
 {
                MessageBox.Show("Match failed");
 }
```

6. Identification.

This is the 1:M matching process in which a matching template is compared with previously stored templates. If the matching score is greater than the threshold, a match is declared. VLMatcher, the matching function from the SDK, is also used for this purpose.

6.1 Initialization of a matcher with the matching template.

```
VLMatcher matcher = new VLMatcher();
matcher.IdentifyStart(template);
```

6.2 The enrolled template(s) is/are read into an array.

```
//Counting stored template files
string[] files = Directory.GetFiles("c:\\frecproj\\frecdata\\templates_db\\permanent\\");
 int count = files.Length;

// Reading all stored templates and storing in an jagged type array

// Declaration of an arrays storedtemplates and sizes
byte[][] storedtemplates = new byte[count][];
int[] sizes = new int[count];

// Looping through all the stored template file(s) and assigning the template(s) to storedtemplates array
for (int i = 0; i < count; ++i)
{
                // Creating a file stream that points to stored template
                 FileStream fs = File.OpenRead(files[i]);

                // Initialization of a binary reader
                BinaryReader reader = new BinaryReader(fs);

                // Assigning the size of the template file to array sizes[] by reading only the integers of the
template file
               sizes[i] = reader.ReadInt32();

                 // Based on the template size create storage space in array and read the content of template
file into the array
               storedtemplates[i] = new byte[sizes[i]];
               reader.Read(storedtemplates[i], 0, sizes[i]);
                // Closing the binary reader.
                reader.Close();
}
```

6.3 The matching process uses a VeriLook matcher (VLMatcher) to declare match or no match.

```
double maxscore = 0;

// Declaration of a scores array to store series of generated scores
double[] scores = new double[count];

// Looping through all the array elements of storedtemplates
for (int i = 0; i < count; ++i)
{
                // Generating score and assigning them to array
                double score = matcher.IdentifyNext(storedtemplates[i]);

                scores[i] = score;

                // Stores the maximum value of score generated in the whole loop process
                if (score > maxscore)
                {
                  maxscore = score;
                }
                score = 0;
}

// Declaration of match or non match based on variable maxscore
if (maxscore > 0.65)
{
                MessageBox.Show("Percentage matched : " + Convert.ToString(Math.Round(maxscore * 100, 1))
+ "%");

 }
else
{
                MessageBox.Show("Match failed");
 }
```

4.3.3 Enrolling and Matching using Video Files

Enrolling and matching using video files involves the images to be extracted from video files in the AVI or WMV format. The images were from video files and used to enroll and match face(s). The following section shows how to extract images from video files and how to enroll and match face(s).

4.3.3.1 Enrollment (videomodeenroll.cs)

1. The enrollment process starts with opening the video file using the OpenFileDialog component of Visual C#.

```
// Initializing the File dialog box component
OpenFileDialog videofile = new OpenFileDialog();

// Filter the file dialog box for AVI and WMV files
videofile.Filter = "Video Files|*.avi;*.AVI;*.wmv;*.WMV";

// Displayes the video selection window
videofile.ShowDialog();
```

2. IMediaDet interface is used to retrieve information about a media file, such as the number of streams, and the media type, duration, and frame rate of each stream (microsoft.com). IMediaDet interface is used to extract frame rate and total duration of the input AVI files. To use this interface, the DLL file named qedit.dll found at C:\windows\system32 must be added to references. After successfully adding the reference to qedit.dll, an entry to DexterLib in the solution explorer window can be seen.

```
// Initialization of imediadet
mdinfo = new MediaDetClass();
mdinfo.Filename = videofile.Filename;

// Calculating total duration of the video file
int len = (int)mdinfo.StreamLength;
```

In the above code, IMediaDet interface is initialized and the total length in seconds of the video file is determined using "mdinfo.StreamLength" function.

3. Frames are extracted from video file.

```
// Get the value of 'from' and 'to' to extract frames from the video using the Text buttons of the form
int startframe;
int endframe;
startframe = Convert.ToInt16(fromframe.Text);
endframe = Convert.ToInt16(toframe.Text);
for (int i = startframe; i <= endframe; i++)
{
```

```
            // For naming templates
            string fBitmapName = i + "-sample" + ".bmp";

            // Extracing bitmap frames and saving the frames in JPEG file format
            md.WriteBitmapBits(i, 640, 480, fBitmapName);
            Image img = Image.FromFile(fBitmapName);
            img.Save(@"c:\\frecproj\frecdata\temp_images\" + i + "-sample" + ".jpg", ImageFormat.Jpeg);
            img.Dispose();

             // Removing the bitmap format images
            System.IO.File.Delete(i + "-sample" + ".bmp");
}
```

Extracting frames from video is achieved using the function method WriteBitmapBits that extracts image frames based on the specified media time (in seconds). The extracted video frames are saved and converted to JPEG file format using "img.Save" function of Visual C#.

4. Compressed and uncompressed templates are created from the extracted images. The uncompressed templates are required since the matcher can only be initialized with uncompressed binary template files.

```
// Counting extracted image frames
string[] files = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_images\\");
 int count = files.Length;

// Creating templates
for (int j = 0; j < count - 1; j++)
{
            // For naming templates with random numbers
            int rn = randnum.Next();
            string fBitmapName = Convert.ToString(rn);

            // Loading and converting frame to gray scale
            NImage image = NImage.FromFile(files[j]);
            NGrayscaleImage grayscale = (NGrayscaleImage)NImage.FromImage(NPixelFormat.Grayscale, 0,
image);
             // Initialization of VeriLook extractor
            VLExtractor extractor = new VLExtractor();

             // Initializing an array structure VleFace[] to detect face(s) to find information of location of
face(s)
             VleFace[] faces = extractor.DetectFaces(grayscale);
             // Face detection using eyes
             for (int i = 0; i < faces.Length; i++)
             {
               VleEyes eyes = extractor.DetectEyes(grayscale, faces[i]);
               byte[] template = extractor.Extract(grayscale, eyes);
               byte[] compressed = extractor.Compress(template);
               string x = rn + "face" + i + j + ".dat";
```

```
            // Storing compressed templates
            FileStream fs = File.Create("c:\\frecproj\\frecdata\\temp_data\\cvideotemplates\\" + x);
            BinaryWriter writer = new BinaryWriter(fs);
            writer.Write(compressed.Length);
            writer.Write(compressed);
            writer.Close();
            fs.Close();
            // Storing uncompressed templates
            FileStream ufs = File.Create("c:\\frecproj\\frecdata\\temp_data\\uvideotemplates\\" + x);
            BinaryWriter uwriter = new BinaryWriter(ufs);
            uwriter.Write(template.Length);
            uwriter.Write(template);
            uwriter.Close();
            ufs.Close();
        }
}
```

The only difference on creating compressed and uncompressed templates is that uncompressed templates are not compressed using SDK function "extractor.Compress".

5. The templates created in step 4 might consist of multiple templates for a single face. Hence, from the pool of templates, the unique template or one template per face is to be separated by deleting the repeating templates of the same face.

The following algorithm that can be applied in this case is:

5.1 The compressed and uncompressed templates are linked to two different linked arrays.

```
string[] ufolder = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_data\\uvideotemplates\\");
string[] cfolder = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_data\\cvideotemplates\\");
```

5.2 The content of the uncompressed and compressed templates are loaded into two different content arrays using C# binaryreader.

```
int mdindex = 0;
int count = 0;

// Do while until the linked template folder was not empty or null
while (ufolder.Length != 0)
{
        // Reading compressed stored templates into an array cstoredtemplates[]
        string[] cfiles = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_data\\cvideotemplates\\");
        int ccount = cfiles.Length;
```

```
byte[][] cstoredtemplates = new byte[ccount][];
int[] csizes = new int[ccount];
for (int i = 0; i < ccount; ++i)
{
   FileStream cfs = File.OpenRead(cfiles[i]);
   BinaryReader creader = new BinaryReader(cfs);
   csizes[i] = creader.ReadInt32();
   cstoredtemplates[i] = new byte[csizes[i]];
   creader.Read(cstoredtemplates[i], 0, csizes[i]);
   creader.Close();
}

// Reading uncompressed template files
string[] ufiles = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_data\\uvideotemplates\\");
int ucount = ufiles.Length;
byte[][] ustoredtemplates = new byte[ucount][];
int[] usizes = new int[ucount];
for (int i = 0; i < ucount; ++i)
{
   FileStream ufs = File.OpenRead(ufiles[i]);
   BinaryReader ureader = new BinaryReader(ufs);
   usizes[i] = ureader.ReadInt32();
   label1.Text = Convert.ToString(usizes[i]);
   ustoredtemplates[i] = new byte[usizes[i]];
   ureader.Read(ustoredtemplates[i], 0, usizes[i]);

   ureader.Close();
 }
```

5.3 The first item of the compressed linked array is initialized. It is moved to the templates folder if it has not been already enrolled. Before moving, the program checks if the template exists for the same person, using VeriLook matcher. If no template exists for the same person, the template is saved.

```
// Check before saving
// Saving compressed templates
// Checking for already enrolled templates
string csourcefile = @cfolder[0];
string cdestination = @"c:\\frecproj\\frecdata\\templates_db\\permanent\\";
string cfilename = Path.GetFileName(csourcefile);

// storing compressed stored enrolled templates into an array estoredtemplates[]
string[] efiles = Directory.GetFiles(@"c:\\frecproj\\frecdata\\templates_db\\permanent\\");
int ecount = efiles.Length;
byte[][] estoredtemplates = new byte[ecount][];
int[] esizes = new int[ecount];
for (int i = 0; i < ecount; ++i)
{
   FileStream efs = File.OpenRead(efiles[i]);
   BinaryReader ereader = new BinaryReader(efs);
   esizes[i] = ereader.ReadInt32();
   estoredtemplates[i] = new byte[esizes[i]];
```

```
    ereader.Read(estoredtemplates[i], 0, esizes[i]);
    ereader.Close();
}

// Use of VeriLook matcher to find out if the similar templates exist.
VLMatcher movematcher = new VLMatcher();
movematcher.IdentifyStart(ustoredtemplates[mdindex]);
for (int m = 0; m < ecount; m++)
{
    double mscore = 0;
    mscore = movematcher.IdentifyNext(estoredtemplates[m]);
    if (maxscore < mscore)
    {
        maxscore = mscore;
    }
}

// Moves only if not found in the existing templates folder
 if (maxscore < 0.65)
{
    File.Move(csourcefile, cdestination + cfilename);
}
else
{
    MessageBox.Show("Ignoring...Template(s) already enrolled in the database");
}
movematcher.Dispose();

    // End of checking and moving code
```

5.4 The first item of the loaded uncompressed template is compared with all the elements of the loaded compressed template array. VeriLook matcher is used for the above comparison. If the matcher generates the score greater than 0.65, then it is identified as a repeating template. Then the template is deleted from the linked folders reloading the linked compressed and uncompressed array as well as compressed and uncompressed templates arrays to reflect the change after deletion of the repeating template. If the score generated by the matcher is less than the threshold, the templates are not deleted. This process is iterated for all the templates until the linked template folders are not empty.

```
VLMatcher matcher = new VLMatcher();
count = ufolder.Length;
matcher.IdentifyStart(ustoredtemplates[mdindex]);
for (int i = 0; i < count; i++)
{
    double score = 0;
    score = matcher.IdentifyNext(cstoredtemplates[i]);
    if (score > 0.65)
```

```
            {
                // Deleting in uncompressed linked folder
                FileInfo deleteufile = new FileInfo(ufolder[i]);
                deleteufile.Delete();

                // Deleting in compressed linked folder
                FileInfo deletecfile = new FileInfo(cfolder[i]);
                deletecfile.Delete();
            }
        }
        matcher.Dispose();

        mdindex = mdindex + 1;

        // Updating the linked folder with new number of templates
        ufolder = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_data\\uvideotemplates\\");
        cfolder = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_data\\cvideotemplates\\");
}
```

4.3.3.2 Matching / Identification (videomodeidentification.cs)

The identification procedures create unique templates and compares with the enrolled template files. Refer step 1 to 4 of section 4.3.3.1 to extract frames from video file and creating templates.

1. Refer 1 of section 4.3.3.1.

2. Refer 2 of section 4.3.3.1.

3. Refer 3 of section 4.3.3.1.

4. Refer 4 of section 4.3.3.1.

5. The unique templates are filtered.

The procedure is:

5.1 The compressed and uncompressed templates are linked in to two different arrays.

```
string[] ufolder = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_data\\uvideotemplates\\");

string[] cfolder = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_data\\cvideotemplates\\");
```

5.2 The uncompressed and compressed templates are loaded into two different arrays.

```
int mdindex = 0;
int count = 0;

while (ufolder.Length != 0)
{

        // Reading compressed stored templates into an array cstoredtemplates[]
        string[] cfiles = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_data\\cvideotemplates\\");
        int ccount = cfiles.Length;
        byte[][] cstoredtemplates = new byte[ccount][];
        int[] csizes = new int[ccount];
        for (int i = 0; i < ccount; ++i)
        {
           FileStream cfs = File.OpenRead(cfiles[i]);
           BinaryReader creader = new BinaryReader(cfs);
           csizes[i] = creader.ReadInt32();
           cstoredtemplates[i] = new byte[csizes[i]];
           creader.Read(cstoredtemplates[i], 0, csizes[i]);
           creader.Close();
        }

        // Reading uncompressed template files
        string[] ufiles = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_data\\uvideotemplates\\");
        int ucount = ufiles.Length;
        byte[][] ustoredtemplates = new byte[ucount][];
        int[] usizes = new int[ucount];
        for (int i = 0; i < ucount; ++i)

        {
           FileStream ufs = File.OpenRead(ufiles[i]);
           BinaryReader ureader = new BinaryReader(ufs);
           usizes[i] = ureader.ReadInt32();
           label1.Text = Convert.ToString(usizes[i]);
           ustoredtemplates[i] = new byte[usizes[i]];
           ureader.Read(ustoredtemplates[i], 0, usizes[i]);
           ureader.Close();
        }
```

5.3 The first element of the uncompressed linked array is moved to unique templates directory.

```
        // Moving uncompressed template files. This is needed for identification purpose
        string sourcefile = @ufolder[0];
        string destination = @"c:\frecproj\frecdata\temp_data\uutemplates\";
        string filename = Path.GetFileName(sourcefile);
        File.Move(sourcefile, destination + filename);
```

5.4 Compare the first item of the loaded uncompressed template with all the elements of the loaded compressed template array. If the matcher generates the score greater than 0.65, it is defined as a repeating template. The repeating template is removed from the linked folders. The linked compressed and uncompressed arrays are reloaded along with the compressed and uncompressed templates arrays. If the score generated by the matcher is less, then none of the templates are deleted. This process is iterated for all templates until the linked template folders are not empty and always moving the first item of the linked array to the unique templates folder after reloading.

```
VLMatcher matcher = new VLMatcher();
count = ufolder.Length;
matcher.IdentifyStart(ustoredtemplates[mdindex]);
for (int i = 0; i < count; i++)
{
   double score = 0;
   score = matcher.IdentifyNext(ustoredtemplates[i]);
   if (score > 0.65)
   {
      // Deleting in uncompressed folder
      FileInfo deleteufile = new FileInfo(ufolder[i]);
      deleteufile.Delete();

      // Deleting in compressed folder
      FileInfo deletecfile = new FileInfo(cfolder[i]);
      deletecfile.Delete();
   }
}

matcher.Dispose();

mdindex = mdindex + 1;


// Updating the linked folder with new number of templates
ufolder = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_data\\uvideotemplates\\");
cfolder = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_data\\cvideotemplates\\");
}
```

6. The identification process uses the enrolled templates from the folder "c:\frecproj\frecdata\templates_db\permanent\" and the matching uncompressed templates from "c:\frecproj\frecdata\temp_data\uutemplates\". In this case all the

uncompressed templates are compared with enrolled templates to find the number of known face(s).

```
// Reading uncompressed template files from folder uutemplates to match to templates of permanent folder
// Output: temputemplates array
string[] tempufiles = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_data\\uutemplates\\");
int tempucount = tempufiles.Length;
byte[][] temputemplates = new byte[tempucount][];
int[] tempusizes = new int[tempucount];
for (int i = 0; i < tempucount; ++i)
{
        FileStream tufs = File.OpenRead(tempufiles[i]);
        BinaryReader tureader = new BinaryReader(tufs);
        tempusizes[i] = tureader.ReadInt32();
        temputemplates[i] = new byte[tempusizes[i]];
        tureader.Read(temputemplates[i], 0, tempusizes[i]);
        tureader.Close();
}

// Reading compressed template files from folder permanent
// Output: permenrollctemplates array

string[] permenrollcfiles = Directory.GetFiles(@"c:\frecproj\frecdata\templates_db\permanent\");
int permenrollccount = permenrollcfiles.Length;
byte[][] permenrollctemplates = new byte[permenrollccount][];
int[] permenrollcsizes = new int[permenrollccount];
for (int i = 0; i < permenrollccount; ++i)
{
        FileStream pcfs = File.OpenRead(permenrollcfiles[i]);
        BinaryReader pcreader = new BinaryReader(pcfs);
        permenrollcsizes[i] = pcreader.ReadInt32();
        label1.Text = Convert.ToString(permenrollcsizes[i]);
        permenrollctemplates[i] = new byte[permenrollcsizes[i]];
        pcreader.Read(permenrollctemplates[i], 0, permenrollcsizes[i]);
        pcreader.Close();
 }

int iresult = 0;
for (int t = 0; t < tempucount; t++)
{
        VLMatcher resultmatcher = new VLMatcher();

        // Matcher initialization one uncompressed template at a time
          resultmatcher.IdentifyStart(temputemplates[t]);
        for (int p = 0; p < permenrollccount; p++)
        {
           double score = 0;
           score = resultmatcher.IdentifyNext(permenrollctemplates[p]);
           label1.Text = Convert.ToString(score);
           if (score > 0.65)
           {
              // Total number of matches
                 iresult = iresult + 1;
           }
        }
        resultmatcher.Dispose();
        // Calculating number of unknow face(s)
        int ufaces = tempucount - iresult;
        // Displaying results
```

```
        MessageBox.Show("Total no. of  known face(s) : " + Convert.ToString(iresult));
        MessageBox.Show("Total no. of  unknown face(s) : " + Convert.ToString(ufaces));

}
```

In the above codes, the filtered templates are loaded into an array (temputemplates) which are compared against the enrolled templates using C# loop and SDK matcher.

7. To preview the frames or identify based upon frame by frame, the tracker component and a picture box component is used in the new form. Under the trackBar_ValueChanged event the following codes are used.

```
private void trackBar1_ValueChanged(object sender, EventArgs e)
 {
      // Declaration of frames array
      string[] frames = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_images\\");

      // Counting the number of frames
      int framecount = frames.Length;

      // Setting the start value of the tracker
      trackBar1.Minimum = 0;

      // Setting the stop or end value of tracker based on the total number of frames
      trackBar1.Maximum = framecount -1 ;

      // Loading the frames in the picturebox
      pictureBox1.Image = Image.FromFile(frames[trackBar1.Value]);
}
```

Frames are loaded into an array using "Directory.GetFiles" function. The length of the frames array is then used to determine the minimum and maximum values of the track bar component. Finally, the picture box component is assigned with the image frame.

8. To view the match and no match face(s) frame by frame, the following process are implied and the results are shown in the preview window. *(videopreview.cs)*

8.1 Images extracted from video files are initialized to an array.

```
string[] frames = Directory.GetFiles("c:\\frecproj\\frecdata\\temp_images\\");
```

8.2 The image frame, which is pointed by the trackbar in the image preview window, is converted to gray scale.

```
// Converting image to grayscale
NImage image = NImage.FromFile(frames[trackBar1.Value]);
NGrayscaleImage grayscale = (NGrayscaleImage)NImage.FromImage(NPixelFormat.Grayscale, 0, image);
```

8.3 The VeriLook extractor is initialized and the face(s) is/are extracted to a VleFace array structure.

```
// Initializing extractor
VLExtractor extractor = new VLExtractor();

// Initializing VleFace[] to detect faces
VleFace[] faces = extractor.DetectFaces(grayscale);
```

8.4 With the help of eyes detection, templates for all the faces are initialized. Then every template is compared with enrolled templates pool to declare match or no match. If there is a match, a blue rectangle is drawn around the matched face otherwise a red rectangle is displayed.

```
for (int i = 0; i < faces.Length; ++i)
{
        // Initialization of VleEyes[] to detect eyes in a face
        VleEyes eyes = extractor.DetectEyes(grayscale, faces[i]);
        // Create template variable with the help of eyes in the found face
        byte[] template = extractor.Extract(grayscale, eyes);

        // //Identification process
        VLMatcher matcher = new VLMatcher();
        matcher.IdentifyStart(template);

        // Counting files
        string[] files = Directory.GetFiles("c:\\frecproj\\frecdata\\templates_db\\permanent\\");
```

```
            int count = files.Length;

            // Reading all stored templates and storing in array
            byte[][] storedtemplates = new byte[count][];
            int[] sizes = new int[count];
            for (int j = 0; j < count; ++j)
            {
              FileStream chfs = File.OpenRead(files[j]);
              BinaryReader reader = new BinaryReader(chfs);
              sizes[j] = reader.ReadInt32();
              storedtemplates[j] = new byte[sizes[j]];
              reader.Read(storedtemplates[j], 0, sizes[j]);
              reader.Close();
            }

           // Initilizaion of graphics contol into the picturebox component
           Graphics g = pictureBox1.CreateGraphics();

          // Finding X,Y , width and height of the detected face
           double rx = faces[i].Rectangle.X + .00;
           double ry = faces[i].Rectangle.Y + .00;
           double rw = faces[i].Rectangle.Width + .00;
           double rh = faces[i].Rectangle.Height + .00;

           // Normalize the coordinates, width and height of detected face for frame size greater than 640x480
           double newrx = (rx / pictureBox1.Image.Width) * 640;
           double newry = (ry / pictureBox1.Image.Height) * 480;
           double newrw = (rw / pictureBox1.Image.Width) * 640;
           double newrh = (rh / pictureBox1.Image.Height) * 480;

           // Default face detection with Red pen. This will convert to blue if the detected face is in the
template db
           Pen pendef = new Pen(Color.Red, 3);
           Rectangle rectdef = new Rectangle(Convert.ToInt32(newrx), Convert.ToInt32(newry),
Convert.ToInt32(newrw), Convert.ToInt32(newrh));
           g.DrawRectangle(pendef, rectdef);

           // Using VeriLook matcher for identification and to generate score
           double score = 0;
           for (int k = 0; k < count; ++k)
           {
             score = matcher.IdentifyNext(storedtemplates[k]);

             // Draws a blue rectangle around the face if the score generated in greater than 0.65
             if (score > 0.65)
             {
               Pen pen = new Pen(Color.Blue, 3);
               Rectangle rect = new Rectangle(Convert.ToInt32(newrx), Convert.ToInt32(newry),
Convert.ToInt32(newrw), Convert.ToInt32(newrh));
               g.DrawRectangle(pen, rect);

               x = Convert.ToString(Math.Round(score * 100, 1)) + "%" + "matched";
               g.DrawString(x, new Font("verdana", 8), new SolidBrush(Color.Blue), Convert.ToInt32(newrx),
Convert.ToInt32(newry));
             }
           }

           // Disposing the initizliaed matcher
           matcher.Dispose();
}
```

4.3.4 Setting up the Matching Threshold

The match or non-match decisions were made according to the defined matching threshold. The score generated by the matcher was compared against the pre-defined matching threshold that was set according to the False Acceptance Rates. The following table illustrates the relationship between matching thresholds and the False acceptance rates (neurotechnologija.com, 2008). FAR represents the chance of allowing unknown face to be accepted. For example, if the matching threshold is 0.625 (62.5%), the FAR will be 1%, meaning that one false face will be accepted as real face out of every 100 faces.

Table 4.1. Threshold value and FAR (neurotechnologija.com, 2008)

False Acceptance Rates (FAR)	Threshold
1 %	0.625
0.1 %	0.650
0.01 %	0.675
0.001 %	0.7
0.0001 %	0.725
0.00001 %	0.750

Defining public threshold variable

In order to set the threshold value that can be used through out the application, the following procedures were implemented.

- Declaring the threshold variable "far" in the public class "farthreshold" on the form "mainform".

```
public class farthreshold
{
        internal static double far;
```

```
}
```

- Setting the threshold variable according to the False Acceptance Rates.

```
// Threshold setup using 1% of FAR
mainform.farthreshold.far = 0.625;
```

- Accessing / Interacting with the threshold value

```
MessgaeBox.Show(Convert.ToString(mainform.farthreshold.far));
```

4.3.5 Managing Template Files

Altogether three folders were used to store template files.

1. c:\frecproj\frecdata\templates_db\temporary\ : Holds the templates that are used for 1:1 Enroll and Verification purpose.
2. c:\frecproj\frecdata\templates_db\permanent\ : Holds the templates that are used for 1:M enroll and identification purpose.
3. c:\frecproj\frecdata\templates_db_video\permanent\ : Can be used to store templates based on video files. In default this folder is not used.

To remove or clear the templates, the following codes were employed using foreach loop and File I/O of C# language.

```
foreach (string tfiles in System.IO.Directory.GetFiles(@"c:\\frecproj\\frecdata\\templates_db\\permanent\\", "*"))
{
        System.IO.File.Delete(tfiles);
}
```

4.3.6 Image and Web Camera Validity Check

- Image Validity Check

The following codes were used to check the resolution and format of images whenever necessary.

```
// Declaring the variable as Image type
public Image eimage;

// Assigning the image file to eimage image type variable from enrollfile.Filename file dialog box
eimage = Image.FromFile(enrollfile.FileName);

// Extract the extension of the image file
string extension = Path.GetExtension(enrollfile.FileName);

// Converting the extension of the image file to lower case
string extension_lower = extension.ToLower();

// Previews the image file in the picture box component if the image is of JPEG type and its minimum
resolution of 640x480
if (((eimage.Width >= 640) & (eimage.Height >= 480)) & ((extension_lower == ".jpg") | (extension_lower ==
".jpeg")))
{
        enrolltextbox.Text = enrollfile.FileName;
        enrollpicturebox.Image = Image.FromFile(enrollfile.FileName);
}
else
{
        MessageBox.Show("Image requirement failed");
}
```

- Web Camera Validity Check

Camera ID property was used to check if the camera is connected to the system. The validation of the images that were captured by the web camera was accomplished using the framewidth and frameheight property of structure CameraVideoFormat. The following codes were used to fulfill the task

```
foreach (Camera camera in cameraMan.Cameras)
 {
        cc = camera.ID;
        activeDevice = camera;
        CameraVideoFormat videoformat = camera.VideoFormat;
        width = videoformat.FrameWidth;
        height = videoformat.FrameHeight;
 }
if (cc != null)
```

```
{
        if ((width < 640) & (height < 480))
        {
          MessageBox.Show("Requirement failed");
        }
}
      else
{
          MessageBox.Show("No web camera is connected");
}
```

4.4 Achieving Enrollment and Matching using FRECPRO

4.4.1 Using the FRECPROJ Application

FRECPROJ application is run using the frec.exe executable file in "C:\frecproj\frecprojrun" folder. Moreover, the project can be run using opening the frec.csproj from the Visual C# development environment by pressing CTRL+F5 keys.

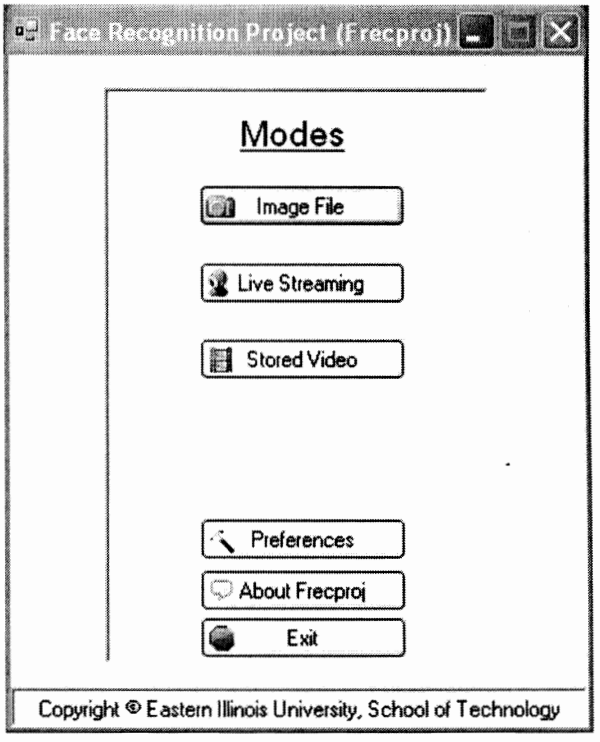

Figure 4.11. Application main window

4.4.2 Image mode

Image mode enrolls and matches face(s) using JPEG files of minimum 640x480 resolution.

4.4.2.1 Verification

In this case an image consisting of a single face is enrolled and the enrolled face is compared with other template to find a match or non-match result. The process is one to one (1:1) since a matching template is compared against an enrolled template.

1. Enrollment

Run the FRECPROJ application → Click Image File → 1:1 Verification → Enroll → Browse the image file → Enroll

2. Matching

Run the application →Click Image File →1:1 Verification → Match →
Browse the image file and the match or non-match result will be displayed with matching percentage

4.4.2.2 Identification

An image consisting of single or multiple faces can be used to enroll images to create a pool of templates. The matching image consisting of single face will be checked against the stored templates to decide a match or no match. This process is 1:M since a matching template is compared against the number of stored templates.

1. Enrollment

Run the application → Click Image File → 1:M Identification → Enroll → Browse the image file → Extract face(s) → After the face(s) is/are detected on the screen, select the template(s) and before enrolling provide a name. The default random name is automatically shown in the rename text box, which can be changed.

Figure 4.12. Multiple faces enrollment

2. Matching

Run the application → Click Image File → 1:M Identification → Match →

Browse the image file and the match or non-match result will be displayed with matching percentage

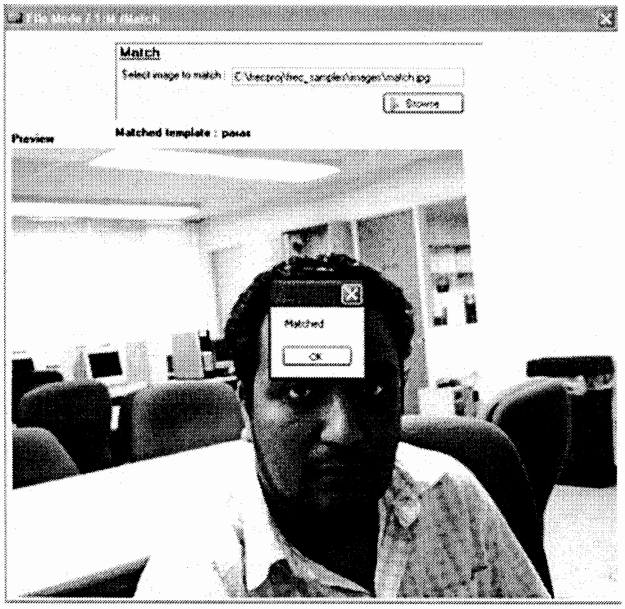

Figure 4.13. Identification

4.4.3 Live streaming mode

Live streaming mode enrolls and matches face(s) using a web camera with the minimum resolution of 640x480.

4.4.3.1 Verification

In this case, live streaming consisting of a single person is enrolled and the enrolled face is matched against other live faces to declare a match or non-match

1. Enrollment

Run the application → Click Live Streaming → 1:1 Verification → Enroll → Enroll to create/store the template → The image frame, which is used to create and store the template, is displayed at the right panel object

2. Matching

Run the FRECPROJ application → Click Live Streaming → 1:1 Verification → Match →

Click verify and the match or non match result will be displayed with matching percentage

4.4.3.2 Identification

Live streaming consisting of single or multiple faces can be used to enroll face(s) to create a pool of templates. While matching, the streaming consisting of single face will be checked against the stored templates to detect match or no match. This process is 1:M.

1. Enrollment

Run the application → Click Live Streaming → 1:M Identification →

Enroll → Extract face(s) → After the face(s) is/are detected, click the template(s) and before enrolling provide a name. The default random name is automatically shown in the rename text box, which can be changed.

Figure 4.14. Enrollment using Live web camera

2. Matching

Run the application → Click Live Streaming → 1:M Identification → Match → Click Identify and the match or non-match result will be displayed with matching percentage

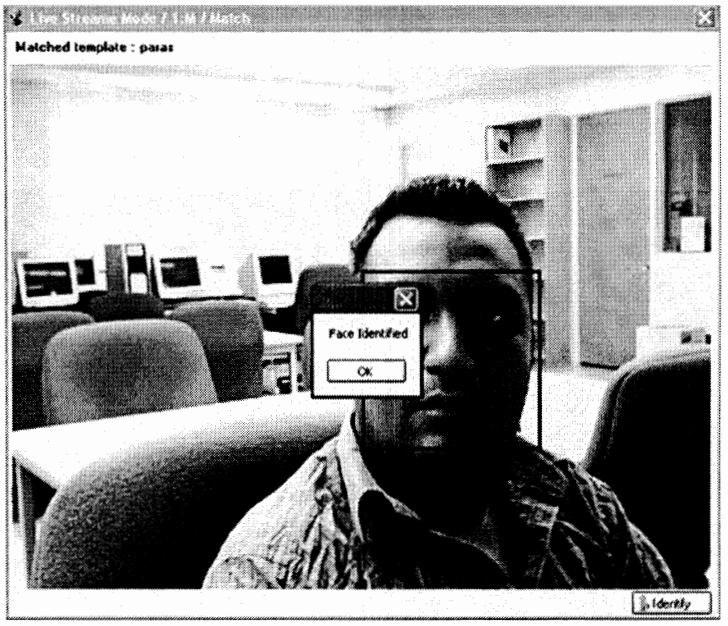

Figure 4.15. Identification in Live streaming web camera

4.4.4 VideoFile Mode

Video file mode enrolls face(s) from AVI and WMV video files. It also can identify face(s) in the video files by using enrolled templates.

4.4.4.1 Identification

A video file is processed to enroll face(s) or to find out how many known or unknown face(s) is/are presented in the video file.

1. Enrollment

Run the application → Click stored video → Enroll → Browse a video file

Notice the total number of frames in the video and type the desired starting frame number and ending frame number → Click Extract → The video segment will be previewed in the bottom left of the window panel which was used to extract face(s) → Click "Enroll template"

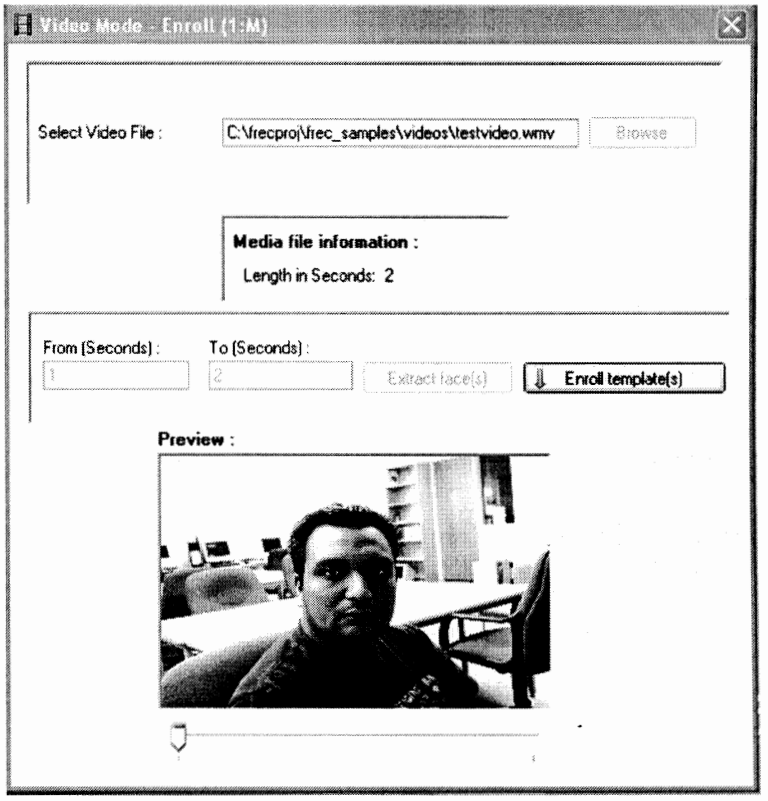

Figure 4.16. Enrollment using video files

2. Matching

Run the application → Click stored video → Match → Browse a video file

Notice the total number of frames in the video and type the desired starting frame number and ending frame number → Click Extract → Press "Identify" to find total number of

known or unknown face(s) or Press "Identify frame by frame" to match frames individually

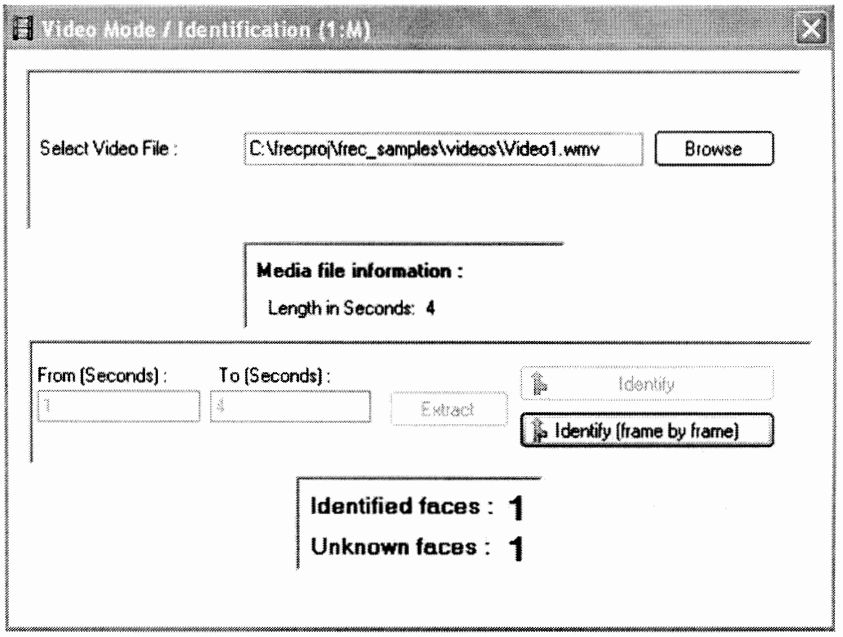

Figure 4.17. Video mode identification

In order identify frame by frame, Click Identify frame by frame → Move the pointer of the trackbar to move between the images frames → Press Identify Frame by Frame button.

Figure 4.18. Identification using frame by frame

4.4.5 Managing preferences

Preferences window can be used to set the threshold value according to the false acceptance rate, delete the stored templates and to open the templates folder.

4.4.5.1 Clear Templates

Clears all templates based on image, live streaming and stored video modes.

4.4.5.2 View Templates Folders

Open the folders, which are used to store templates. The folder named temporary hold the templates that are used for verification purposes and the folder named permanent holds the pool of templates used for identification purposes.

4.4.5.3 Setting threshold value

To set the threshold, select the desired thresholds according to the false acceptance rate and press the OK button.

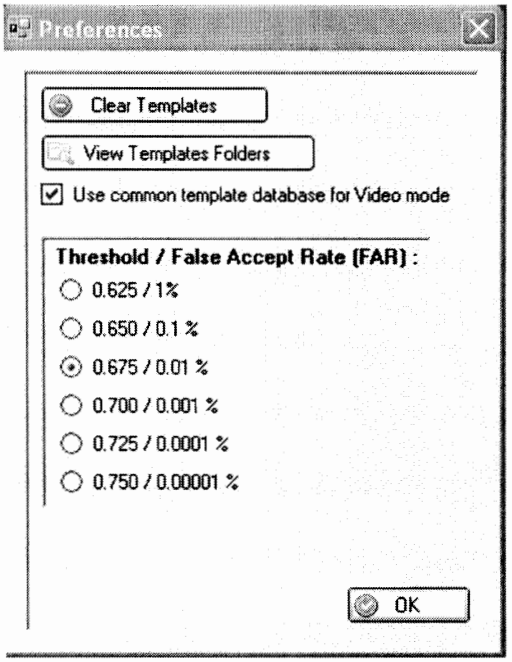

Figure 4.19. Preferences windows

Chapter 5

Discussion

The basic function of any biometric face recognition systems involves enrolling and identifying human faces, which are dependent on intelligent algorithms. It would be a tremendous task for the face recognition system programmers to develop the algorithms and also to integrate them to develop face recognition applications. In other words, developing algorithms and the face recognition systems can be divided into two (2) different areas for software development process.

To facilitate the development or integration of face recognition systems, software development kit (SDK) has been developed. SDK provides pre-built functions and libraries to be reused by other programmers of face recognition systems. The challenge is to find ways of reusing the existing library of functions provided by SDK while developing new face recognition systems. The focus of this research is to provide experience and knowledge of integrating SDK in a typical system development of face recognition applications, using VeriLook, an available SDK. The experience and knowledge gained from this research will help future developers to efficiently integrate SDK for their system development.

5.1 Characteristics of Face Recognition SDK

SDK allows its functions and libraries to be reused by the development environment such as Visual C#. Some of the characteristics of SDK are:

- It acts as an additional layer, which provides added functionalities to the development environment with the help of DLL files that can be referenced inside

the development environment.

- It simplifies the development of face recognition application development by providing functions and libraries incorporating complex algorithms of face detection and recognition. In other words, a system developer does not have to deal with face recognition algorithms.
- It can be integrated into various application development environments such as: Visual C#, Visual C++ and Visual Basic.NET.
- It provides functionalities for camera management, image processing, face detection, template generation, and matching in the form of verification and identification.

5.2 Effective Procedure for Developing Face Recognition Systems

In order to develop face recognition software application, the following procedures were identified.

5.2.1 Choosing of an SDK

Various face recognition SDKs are available today. Some provides 1:1 enrollment and matching only where as some provides 1:M for enrolling and matching. Thus, SDK needs to be chosen according to the need of the developers and customer requirements. VeriLook SDK provides all of the required functions for the development of face recognition systems such as enrolling and matching human faces in 1:1 and 1:M modes, camera initialization, camera management and image processing.

5.2.2 Determining possible development environments

Face SDK runs on different development environment such as Visual C#, Visual C++ and Visual Basic.NET. Hence, any one platform must be chosen according to the expertise of the developers and the nature of the development environment. It is found that Microsoft Visual C# Express Edition which can be downloaded without any charge is sufficient to work with VeriLook SDK in order to develop face recognition applications.

5.2.3 Analyzing hardware and software requirements

Hardware resources must be carefully analyzed to set up the development environment since the .NET based development tools consumes a lot of memory and CPU. Also the operating system platform must be carefully chosen based on the experiences of the developers. In this research, a Pentium 4 machine with 1GB of RAM was sufficient to support Window XP operating system platform.

5.2.4 Setting up of a machine for development with the preferred operating system

After the operating system is installed, the Operating system environment must be updated, so that problems do not occur while development. The system can be updated using the update programs shipped with the operating system. In case of windows, "Windows update" from the operating system's control panel can be used.

5.2.5 Setting up the development environment

The development tool such as Visual C# Express Edition can be downloaded and installed into the machine.

5.2.6 Purchase/Download the SDK.

5.2.7 Installing the SDK

Refer to the installation manual to properly install the SDK.

5.2.8 Adding references of the SDK to the development environment

After the successful installation of SDK, the DLL files provided by the SDK must be referenced inside the development environment. In case of VeriLook, the DLL files that must be referenced are:

1. Neurotech.dll,
2. Neurotech.Images.dll
3. Neurotech.Cameras.CameraMan.dll
4. Neurotech.Biometrics.VLExtractor.dll
5. Neurotech.Biometrics.VLMatcher.dll

5.2.9 Start the development process.

After the SDK has been properly referenced, application development can be started. Usually the SDK manuals, tutorials and documentation provide resources in development.

Improvements for VeriLook SDK

- The documentation on the structure of templates generated by VeriLook SDK is not available or is vendor locked. Hence for vender independence and interoperability, the structure of the templates must be provided to the interested

developers.

- Since VeriLook does not support enrollment and matching directly using video files, the images should be extracted from video files and use them to enroll and match. To facilitate easier implementation, VeriLook SDK should support enrollment and matching using video files
- VeriLook SDK is very sensitive to the input images it uses to create templates. If an image file is corrupted or has minor problem, then the templates are not created. Hence, there should be self-detection and recovery for such kind of images.
- The VeriLook matcher can only be initialized with uncompressed binary templates. Thus, the SDK matcher should also support compressed templates at the time of matcher initialization. This would reduce the complexities in working with the templates generated by VeriLook.
- To run the applications developed using VeriLook SDK requires trial version of SDK or licensed SDK. It would be better if the vendor provides run time libraries without any charge.

Chapter 6

Conclusion

Face recognition systems are implemented to secure facilities and access controls, which is one of the promising biometric security technologies. This research explored the process of face recognition system development using VeriLook SDK. It was aimed at assisting future developers to use biometric face recognition SDK for future integration and development.

A prototype named FRECPROJ was developed, to illustrate the technical aspects on the development of face recognition systems. Developers and integrators may use the sample project as a reference for the development of their own face recognition systems. The prototype can enroll, verify and identify human faces. Various input sources were studied and implemented in the prototype for enrollment and matching, including a still image file, live streaming from web camera, or a video file in AVI or WMV format.

The advantages on using a face recognition SDK is that, it adds biometric functions and libraries to development environment so that developers and integrators can utilized them as they use normal programming functions and methods.

Developers and integrators then effectively utilize SDK to shorten the development cycle time. The disadvantages of using SDK are related to the fact that feature rich SDK must be purchased and their source codes are proprietary.

To conclude, VeriLook SDK provides an additional layer of libraries to the development environment in order to develop face recognition systems. The SDK can be purchased and integrated with a number of development tools such as Visual C# and Visual C++. After the libraries are referenced to the development environment, the face

recognition functions and methods are available within the development environment. By conducting this research, the details of biometric face recognition system development are understood. It is understood that SDK eases the development process without undergoing into the development of complex algorithms of face recognition. It is hoped that this research will assist the future programmers in the development of face recognition systems.

APPENDIX

myvideo.cs

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Windows.Forms;

namespace Neurotec.Biometrics.frec
{
   public class VideoControl : System.Windows.Forms.Panel
   {

      public string ivmode;

      // Sets the value of image
      private Bitmap image;

      public Bitmap Image
      {
         get
         {
            return image;
         }
         set
         {
            image = value;
         }
      }

      // Sets the faces which contains the co-ordinates of detected faces
      private VleFace[] faces;

      public VleFace[] Faces
      {
         get
         {
            return faces;
         }
         set
         {
            faces = value;
         }
      }

      // Returns detection details
      private VleDetectionDetails detectionDetails;

      public VleDetectionDetails DetectionDetails
      {
         get
         {
```

```
            return detectionDetails;
        }
        set
        {
            Invalidate();
        }
    }

    protected override void OnPaint(PaintEventArgs pe)
    {
        Graphics g = pe.Graphics;
        if (image != null)
        {
            // Draws the image passed from the form codes
            g.DrawImage(image, 0, 0, 640, 480);
        }

        // Draws rectangle
        if ((faces != null) && (faces.Length != 0)) // Checks if face(s) is/are available
        {
            Pen pen = new Pen(Color.Blue, 3);

            if (ivmode == "ver")
            {
                g.DrawRectangle(pen, faces[0].Rectangle);
            }

            if (ivmode == "idn")
            {

                for (int i = 0; i < faces.GetLength(0); ++i)
                {
                    g.DrawRectangle(pen, faces[i].Rectangle);
                }
            }

        }
    }

    // This is necessary to prevent the background from being painted.
    protected override void OnPaintBackground(PaintEventArgs pevent)
    {
        //do nothing
    }
  }
}
```

References

An Introduction to Biometrics - Face Recognition. (2008, January). Retrieved January 29, 2008, from http://www.tiresias.org/guidelines/biometrics_face.htm

Animetrics90 Product Family Overview. (2007). Retrieved March 22, 2008, from http://www.animetrics.com/products/

Authentication definition. (n.d.). Retrieved March 1, 2008, from http://www.pcmag.com/encyclopedia_term/0,2542,t=authentication&i=38192,00.asp

Authentication definition. (2006, January). Retrieved March 22, 2008, from http://www.bellevuelinux.org/authentication.html

Authorization definition. (n.d.). Retrieved March 1, 2008, from http://www.pcmag.com/encyclopedia_term/0%2C2542%2Ct%3Dauthorization&i%3D38202%2C00.asp

BioID SDK. (2006). Retrieved March 22, 2008, from http://www.humanscan.de/products/bioid/index.php

Biometrics FAQ. (2008, January). Retrieved January 29, 2008, from http://www.bromba.com/faq/biofaqe.htm

Biometrics Glossary. (2006, February). Retrieved January 30, 2008, from www.biometricscatalog.org/biometrics/GlossaryDec2005.pdf

Biometric SDK. (2005). Retrieved January 29, 2008, from http://sourceforge.net/projects/biometricsdk

Biometrics Standards. (2006, August). Retrieved November 7, 2007, from http://www.biometrics.gov/Documents/BioStandards.pdf

Biometric-Watch - Biometric Vendor Listings. (n.d). Retrieved January 28, 2008, from http://biometricwatch.com/vendors.htm

Bolton, D. (n.d.). *Definition of .NET.* Retrieved January 29, 2008, from http://cplus.about.com/od/introductiontoprogramming/g/dotnetdefn.htm

Bromba, M. (2007, January). *Biometric Myths*. Retrieved February 28, 2008, from http://www.bromba.com/knowhow/biomyths.htm#degrees

BS ISO/IEC 19794-5:2005 Information technology. Biometric data interchange formats. Face image data. (n.d.). Retrieved November 28, 2007, from http://www.bsi-global.com/en/Shop/Publication-Detail/?pid=000000000030107738

Case studies. (2008). Retrieved March 22, 2008, from http://www.neurotechnologija.com/cgi-bin/customers.cgi

Chan, W. (2006, August). *NIST releases open-source kit for biometrics.* Retrieved January 29, 2008, from http://www.fcw.com/online/news/95498-1.html

DLL. (n.d.). Retrieved March 1, 2008 from http://www.webopedia.com/TERM/D/DLL.html

FaceIt® SDK. (2008). Retrieved March 22, 2008, from http://www.l1id.com/index.php?option=com_content&task=view&id=204&Itemid=184

FaceVACS-SDK. (n.d). Retrieved March 22, 2008, from http://www.cognitec-systems.de/products-sdk.htm

Face biometrics (n.d.). Retrieved March 1, 2008 from http://www.biometricnewsportal.com/face_biometrics.asp

Hodgson, J. (1996, October). *Software Engineering Basics*. Retrieved January 30, 2008, from http://www.sju.edu/~jhodgson/se/softeng.html

ImediaDet Interface. (2008). Retrieved May 1, 2008 from http://msdn.microsoft.com/en-us/library/ms785892(VS.85).aspx

Information technology - Biometric data interchange formats. (2005, June). Retrieved November 27, 2007 from http://webstore.iec.ch/preview/info_isoiec19794-5%7Bed1.0%7Den.pdf

ISO/IEC 19794-5:2005. (n.d.). Retrieved November 28, 2007, from http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=38749

Jain, A, K., Ross, A., & Prabhakar, S. (2004, January). *An Introduction to Biometric Recognition*. Retrieved January 29. 2008, from htp://biometrics.cse.msu.edu/Publications/GeneralBiometrics/JainRossPrabhakar_BiometricIntro_CSVT04.pdf

JockerSoft. (2006, March). *Extract frames from video files.* Retrieved March 1, 2008,from http://www.codeproject.com/KB/graphics/ExtractVideoFrames.aspx

Miller, M. (2004, January). *The importance of Standards*. Retrieved Decemeber 7, 2007, from http://www.energyrating.gov.au/pubs/2004ac2plen-miller.pdf

MIRH Eye SDK. (n.d) Retrieved March 22, 2008, from http://www.dreammirh.com/english/products/eyeseries.html

Nanavati, S., Thieme, M., & Nanavati, R. (2002). *Biometrics: Identity Verification in a Networked World*. Wiley Tech

Open Source Computer Vision Library. (n.d.). Retrieved March 1, 2008 from http: //www.intel.com/technology/computing/opencv/

Pissarenko, D. (2002). *Eigenface-based facial recognition.* Retrieved March 22, 2008, from http://openbio.sourceforge.net/resources/eigenfaces/eigenfaceshtml/facesOptions.html

Published American National Standards Developed by INCITS M1 - Biometrics. (2007, April). Retrieved November 7, 2007 from http://www.itl.nist.gov/div893/biometrics/documents/April%206_%20FP_Published_INCITS_M1_Standards.pdf

Ratha, N, K., Connell, J, H., & Bolle, R, M. (2001, April). *Enhancing security and privacy in biometrics-based authentication systems.* Retrieved March 1, 2008 from http://www.research.ibm.com/journal/sj/403/ratha.html

Rehman, R,U., & Paul, C. (2003). *Introduction to Software Development.* Retrieved January 28, 2008, from http://www.faqs.org/docs/ldev/0130091154_24.htm

Ronkkonen, J. (n.d.). *Video Based Gait Analysis in Biometric Person Authentication: An Brief Overview.*

Ross, S, T. (1999). *Computer Security: A Practical Definition.* Retrieved January 27, 2008, from http://www.albion.com/security/intro-4.html#pgfId-449937

SDK. (n.d.). Retrieved March 1, 2008, from http://isp.webopedia.com/TERM/S/SDK.html

Security. (n.d.). Retrieved December 7, 2007, from http://www.thefreedictionary.com/security

Shao, J. (2005). *Implementation of an Open Source Enterprise Resource Planning (ERP) System.* Charleston, IL: Eastern Illinois University

Software Developers' Kit. (2008). Retrieved March 22, 2008, from http://www.acsysbiometrics.com/product_sdk.html

Software Engineering. (n.d.). Retrieved January 29, 2008, from http://www.webopedia.com/TERM/S/software_engineering.html

Swaroop, C, H. (2005). *The Software Development Process.* Retrieved January 29, 2008, from http://www.ibiblio.org/g2swap/byteofpython/read/software-development-process.html

Technical - Security (Biometrics - False Accept/Reject). (2004). Retrieved January 26, from http://www.farpoint.com/false_ar.htm

Tilton, C. (2006, January). Biometric Standards-An Overview. Retrieved January 2008, from www.daon.com/downloads/standards/Biometric%20Standards%20White%20Paper%20Jan%2006.pdf

VeriLook Face Identification SDK. (2008). Retrieved March 1, 2008, from http://www.neurotechnologija.com/vl_sdk.html

VeriLook, PC-based Face Recognition Technology. (2008). Retrieved January 29, 2008, from http://www.neurotechnologija.com/verilook.html

VeriLook 3.0 SDK. (2007, June). Retrieved January 26, 2008, from http://www.neurotechnologija.com/download/VeriLook_3_1_Standard_SDK_Trial.zip

Visual Studio .NET. (n.d.). Retrieved March 1 , 2008, from http://www.pcmag.com/encyclopedia_term/0,2542,t=Visual+Studio+NET&i=54003,00.asp

Visual Studio Express Editions. (2007). Retrieved March 1, 2008, from http://www.microsoft.com/express/default.aspx

What is an API ? (n.d.). Retrieved March 1, 2008, from http://www.querycat.com/search?q=API%20+%20set%20of%20routines,%20protocols%20and%20tools

What is biometrics ?. (n.d.). Retrieved March 1, 2008, from http://www.webopedia.com/TERM/B/biometrics.html

Woodward, J, D., Orlans, N,M., & Higgins, P, T. (2003). Biometrics. Mc-Graw-hill

XID Technologies Introduction. (2008). Retrieved March 22, 2008, from http://www.xidtech.com/About%20Us.htm

Youseef, M. (2007, February). C# Delegates Explained. Retrieved May 1, 2008, from http://www.aspfree.com/c/a/C-Sharp/C-Sharp-Delegates-Explained/